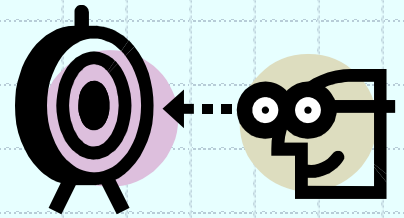
Association Rule Mining in Peer-to-Peer Systems

Assaf Schuster

Joint work with Ran Wolff

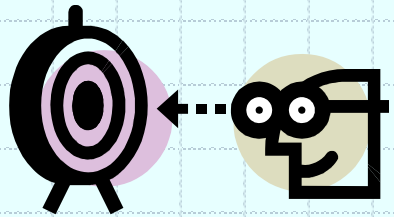
ICDM 2003

System-Centric View of P2P



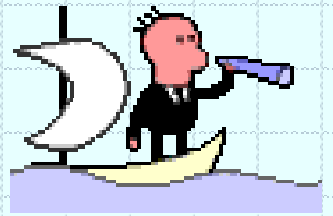
- Internet scale system
- Failure rate
  - high as it has ever been
- Communication
  - sparse and slow, especially upstream
- Supported features
  - Local failure detection
  - Overlay network maintenance
  - Routing

Data-Centric View of P2P



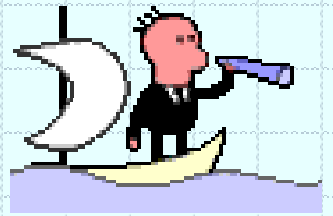
- Horizontally distributed database
- Impossible to collect – upstream comm.
- Data changes rapidly
  - Partitions disappear and new ones are added
  - Data is modified faster than it can be propagated

Data-Mining a P2P System

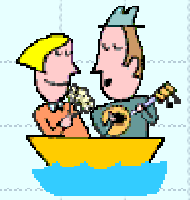


- Impossible to collect the data
  - Distributed algorithms
- Ever changing data and system
  - Incremental algorithms
  - Ad hoc, anytime results
- Internet scale
  - No global operators allowed

Why Mine a P2P Database?



- Find out that eDonkey users who read anti-terrorism texts also download stock manipulation software!
- Find out that Kazaa Matrix fans also like Radiohead music!



Kazaa

- 60M users
- 45M files downloaded/month
- 5M simultaneously connected users
- 900M shared files
- Downloaded 229M times

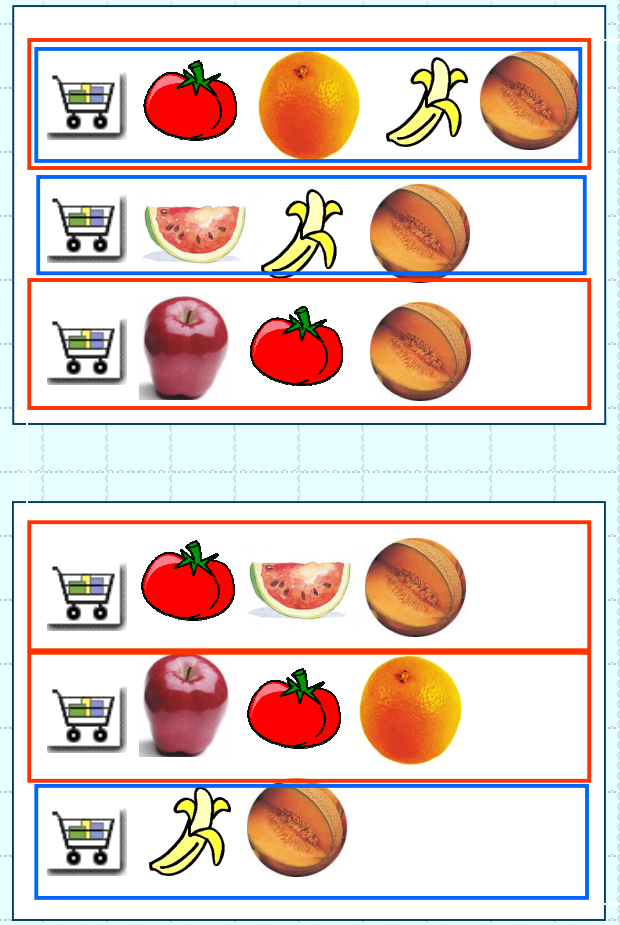
Large-Scale Distributed Example: Parallelization

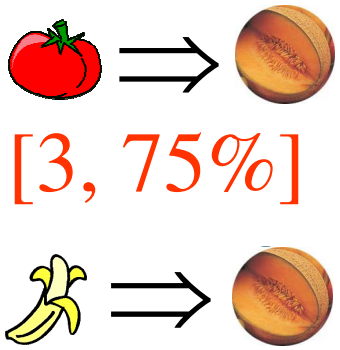
- Technion has ~10,000 PCs
  - Avg. Memory – 100MB
  - Connected via 100Mb/Sec Ethernet
- To mine a 1TB database
  - Read it once and pour it into the memory of 10,000 PCs (disk IO bounded operation)
  - Build your decision tree / association rules
- The alternative is out-of-core mining

LSD Example: Next-Gen Monitoring

- In the Concurrent & Distributed Programming course students implement a parallel program using MPI and run it on a Condor pool
- It took us about a week to find out the following pattern:
  - an MPI process accesses a dll
  - it fails to find it
  - exits abnormally
  - condor assumes the job was evicted and reruns the whole MPI program
- Distributed association rule mining would have found this pattern immediately

Association Rule Mining





MinFreq = 3
MinConf = 70%

ARM Examples

- Diapers ➔ Beer
- Diabetics + Heart problems
- The Remedia case

BUT…PRIVACY!

Our Work

- Association rule mining algorithm for peer-to-peer systems
  - ☆ **Local** and therefore infinitely scalable
  - ☆ **Asynchronous** and therefore fast
  - ☆ **Dynamic** and therefore robust
  - ☆ **Accurate** – not approximated
  - ☆ **Anytime** – you get early results fast

Related Work

- "Estimating aggregates on a peer-to-peer network". M. Bawa, H. Garcia-Molina, A. Gionis, and R. Motwani. TR-2003
- "Gossip-Based Computation of Aggregate Information", D. Kempe, A. Dobra, and J. Gehrke. PODC-2003
- "Communication-efficient Distributed Mining of Association Rules". A. Schuster, R. Wolff. SIGMOD-2001

Other related work

- Assumes uniform sources
- Data independent
- Estimated
- Global
- Assumes static data

NON SCALABLE!!!

Distributed ARM algorithm

1. Reducible to a series of **majority votes** among transactions
   - Itemset X:
     - Vote 1 if they contain the itemset, 0 otherwise
     - MinFreq majority required
   - Rule X⇒Y:
     - Vote 1 if they contain X and Y, 0 if they contain only left-hand side, the rest abstain.
     - MinConf majority required

Distributed ARM algorithm

2. Majority vote can be calculated by a **local** algorithm
   - Local algorithms: a peer consults few nearby neighbors to calculate result
3. Candidate generation – generalized for on-the-fly rule generation

Solution to Traditional ARM

$$Let\ 0 < MinFreq \le 1, 0 \le MinConf \le 1$$

$$R[DB] = \left\{ X \Rightarrow Y : \begin{array}{c} X \cap Y = \phi \\ Freq(X \cup Y, DB) \ge MinFreq \\ Freq(X \cup Y, DB) \ge MinConf \cdot Freq(X, DB) \end{array} \right\}$$

Difference in P2P ARM

$$DB \rightarrow DB_t^u : \forall u, t$$

$$R[DB_t] = R\left[\bigcup_v DB_t^v\right]$$

Solution for P2P-ARM

- No termination
- Anytime solution
  - The solution of peer u at time t : $\tilde{R}_u[DB_t]$
  - The global solution : $R[DB_t] = R\left[\bigcup_v DB_t^v\right]$
- Quality judged by *recall* and *precision*

$$\text{Recall} = \frac{\left|\tilde{R}_u[DB_t] \cap R[DB_t]\right|}{\left|R[DB_t]\right|}$$

$$\text{Precision} = \frac{\left|\tilde{R}_u[DB_t] \cap R[DB_t]\right|}{\left|\tilde{R}_u[DB_t]\right|}$$

Local Majority Voting algorithm

- Messages are pairs $<s,c>$
  - Number and sum of the bits
- Each peer u stores the last message it sent to and received from every neighbor v – $<s^{uv}, c^{uv}>$ and $<s^{vu}, c^{vu}>$

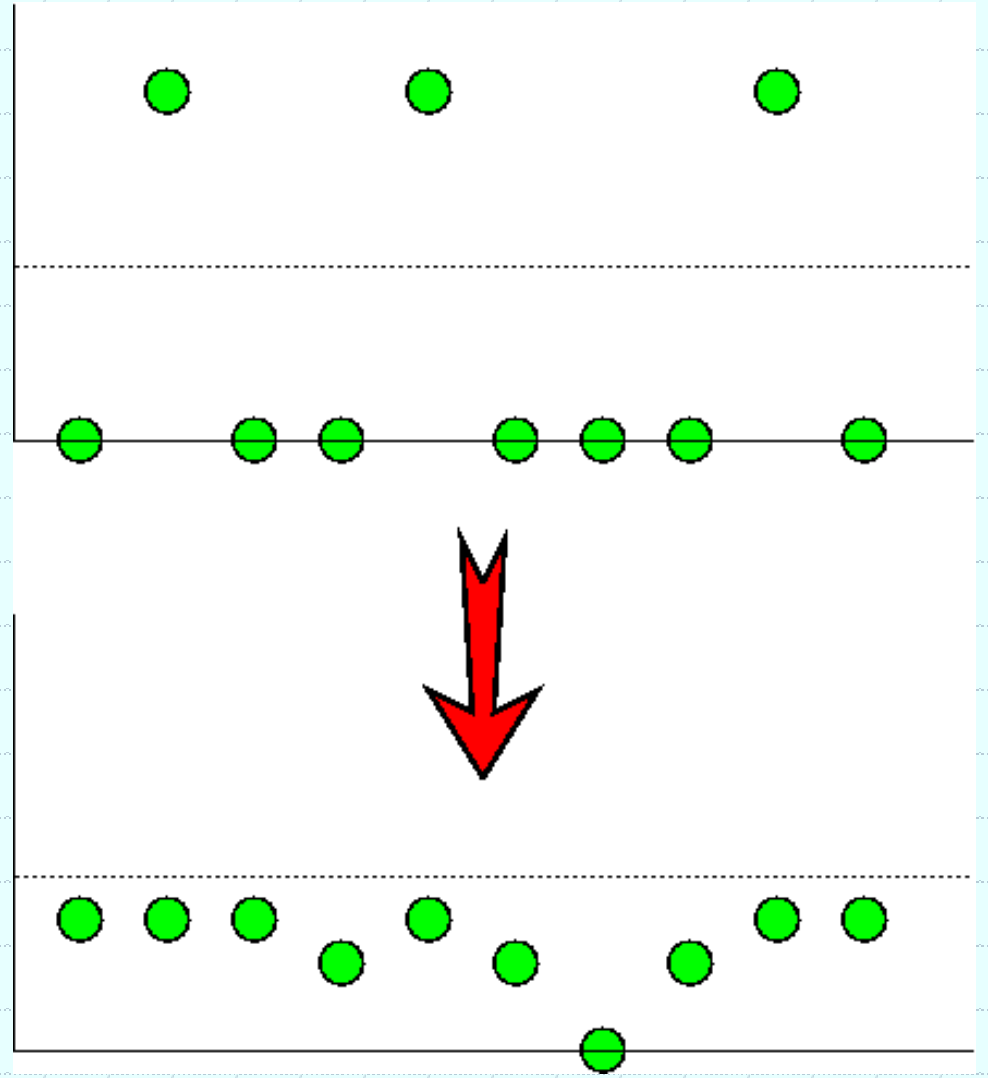
$$\Delta^{uv} = \left(s^{vu} + s^{uv}\right) - \lambda\left(c^{vu} + c^{uv}\right)$$

$$\Delta^{u} = \sum_{v \in N_t^u} \left(s^{vu} - \lambda c^{vu}\right)$$

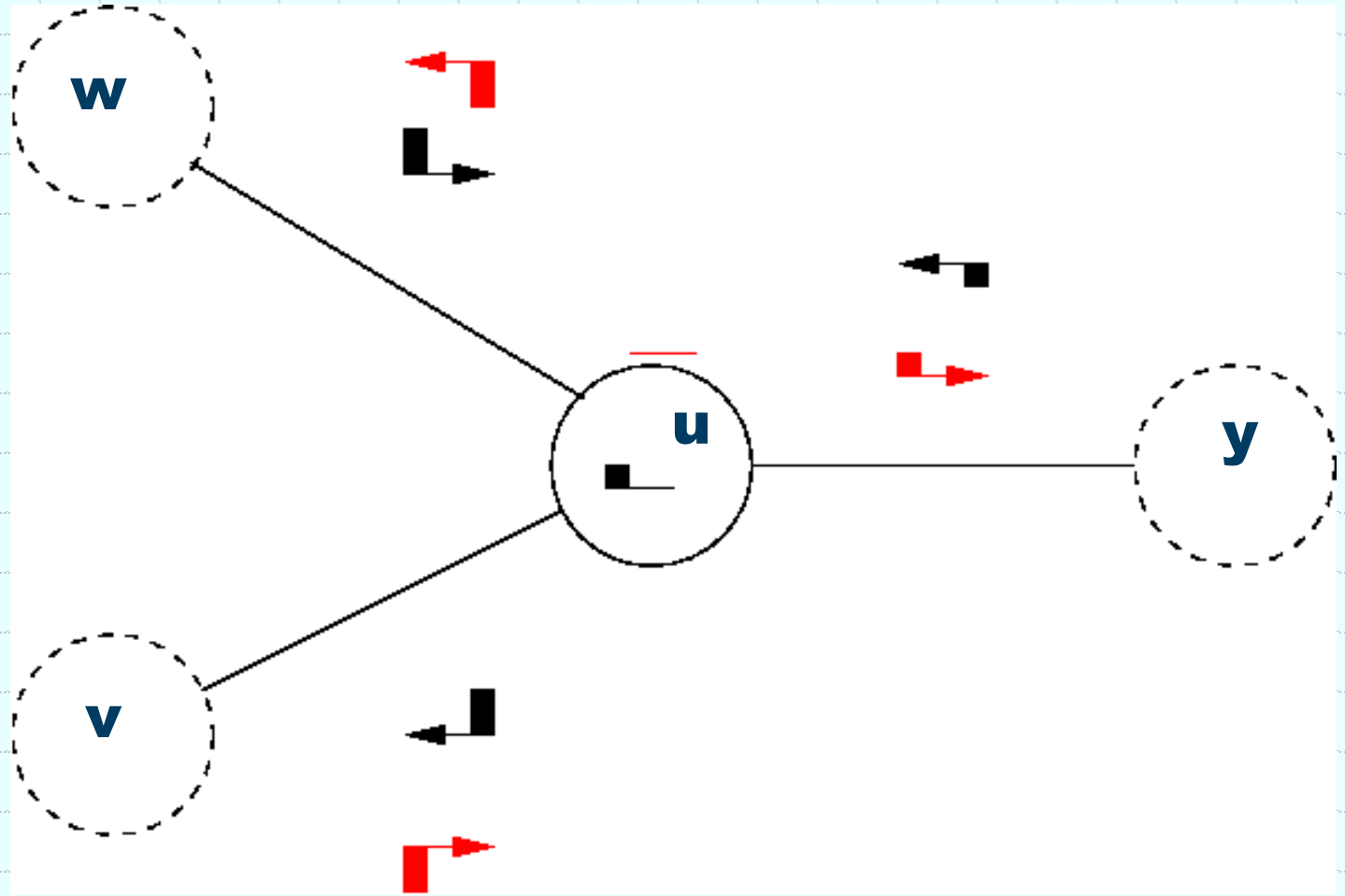
Local Majority Voting algorithm

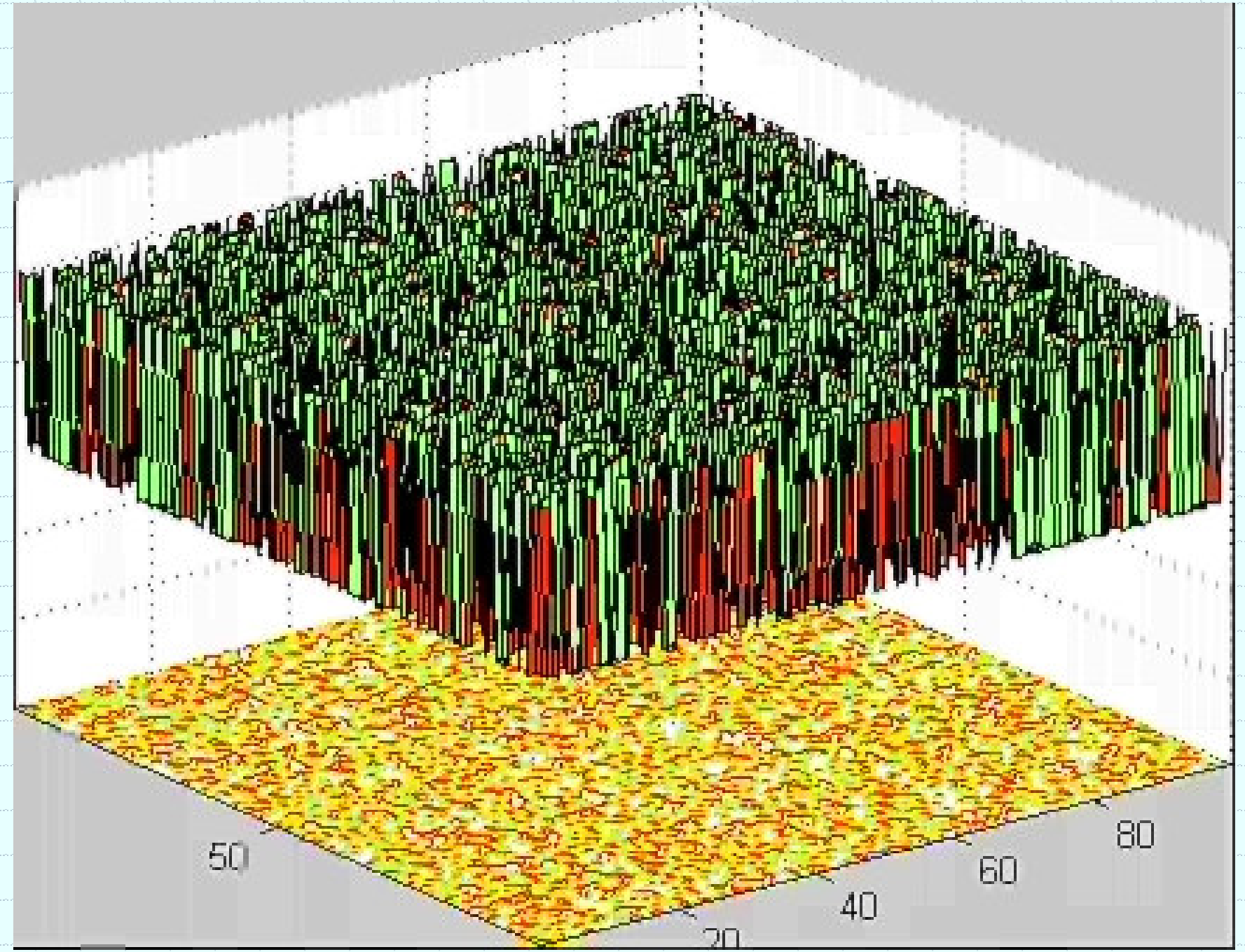
- Δ^u – peer u evidence for the number of excess 1 votes
- Δ^{uv} – peers u and v agreement on the evidence for the number of excess 1 votes
- Main idea:
  - Have each peer agree with its neighbors on Sign(Δ^u)
  - Make sure no peer misleads its neighbors
  - $\Delta^{uv} \geq 0$ and $\Delta^{uv} > \Delta^u$ or $\Delta^{uv} < 0$ and $\Delta^{uv} < \Delta^u$

Local Solution for a Majority Vote



LSD-Majority – Example




50
40
60
80
20

Candidate Rule Generation

- Initial itemsets

$$\text{For each } i \in I$$

$$\mathrm{C} \leftarrow \langle \phi \Rightarrow \{i\}, MinFreq \rangle$$

- Initial rules

$$\text{For each } \langle \phi \Rightarrow X, MinFreq \rangle \in \tilde{R}_u[DB_t]$$

$$C \leftarrow \left\{ \langle X \setminus \{i\} \Rightarrow \{i\}, MinConf \rangle \mid i \in X \right\}$$

Candidate Rule Generation

- AprioriGen

$$\text{For each } X \cup \{i_1\}, X \cup \{i_2\} \in L - \textit{Set of frequent itemsets}$$

$$\text{Verify } \forall_{i_3 \in X}\ X \cup \{i_1, i_2\} \setminus \{i_3\} \in L$$

$$C \leftarrow X \cup \{i_1, i_2\}$$

- Generalized AprioriGen

$$\text{For each } \langle X \Rightarrow Y \cup \{i_1\}, \lambda \rangle, \langle X \Rightarrow Y \cup \{i_2\}, \lambda \rangle \in \tilde{R}_u[DB_t]$$

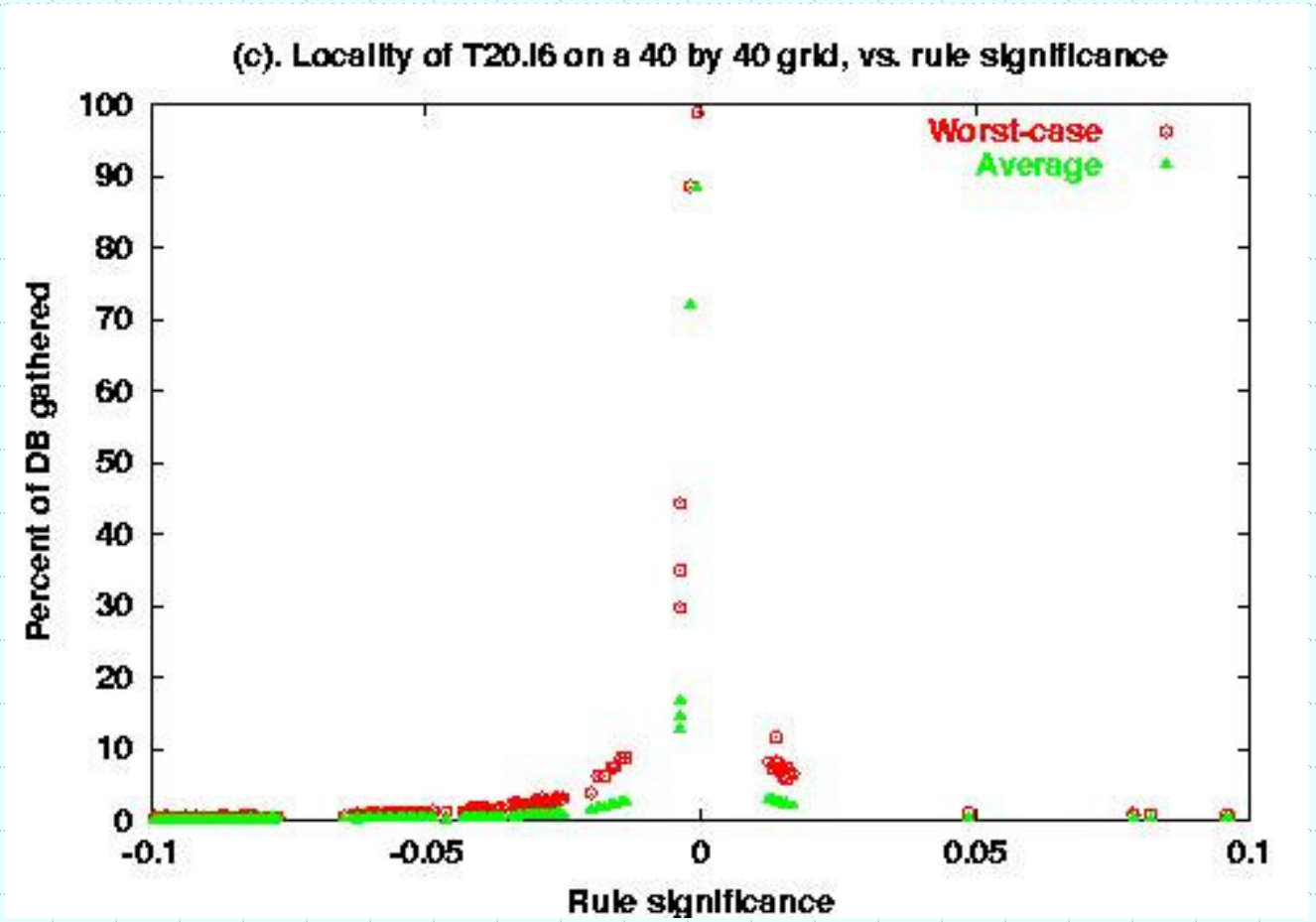
$$\text{Verify } \forall_{i_3 \in Y} \langle X \Rightarrow Y \cup \{i_1, i_2\} \setminus \{i_3\}, \lambda \rangle \in \tilde{R}_u[DB_t]$$

$$C \leftarrow \langle X \Rightarrow Y \cup \{i_1, i_2\}, \lambda \rangle$$

Summary

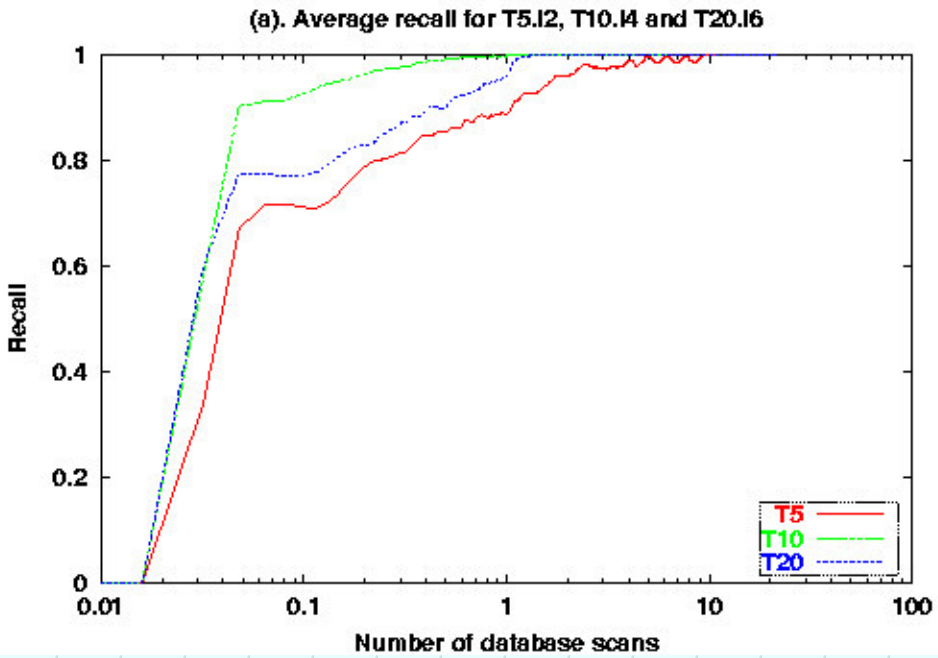
- Each peer
  - Creates initial itemsets
  - Repeats forever
    - Count the local support of each candidate
    - Participate in a majority vote for each candidate
    - Update the local solution
    - Create new candidates based on the local solution

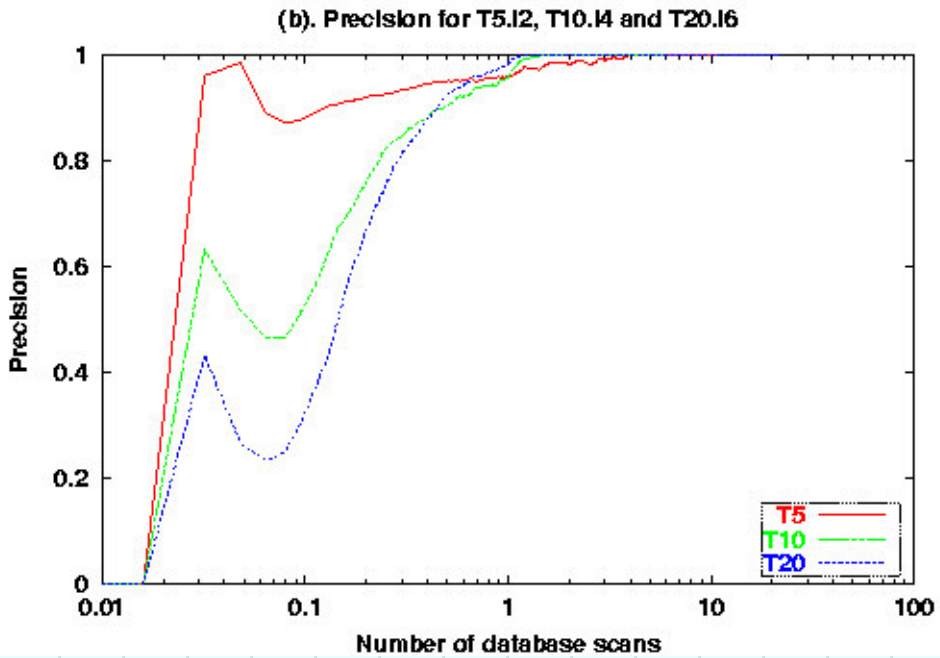
Locality



- 1,600 peers
- All initiated at once
- Local DB of 10K transactions
- Locked step
- Run until there are no further messages

Convergence

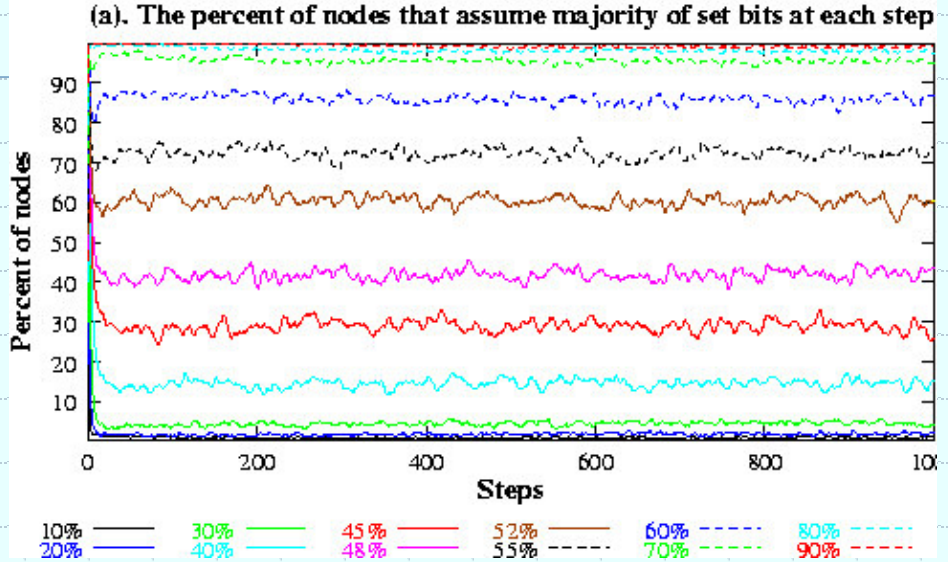




By the time the database is scanned once, in parallel:

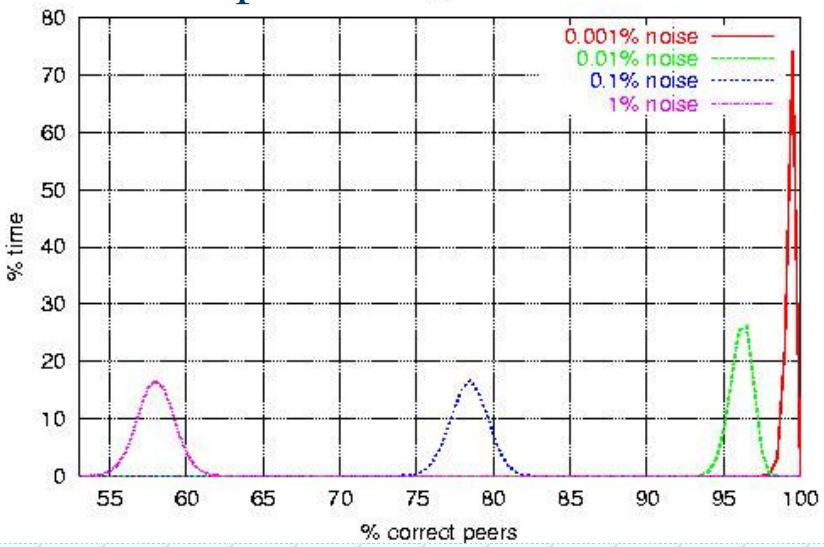
- the average node has discovered 95% of the rules;
- has less than 10% false rules.

Dynamic Behavior – 1M-peers

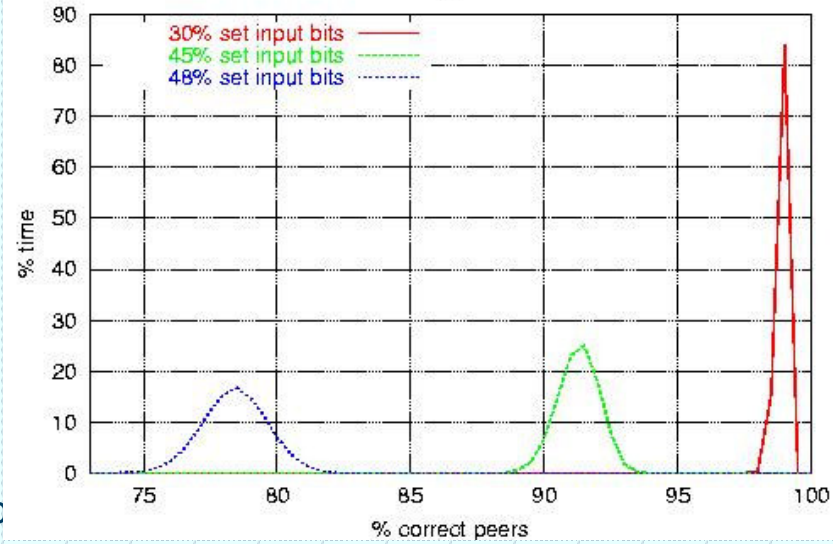


1% noise

48% set input bits



0.1% noise



e and
24 Nc

Thank you

Relation to Other Domains

- Local algorithms – the persistent bit problem
- Distributed system – diffusive load balancing

Future work

- Privacy preserving
- General communication graphs
- Other data mining problems