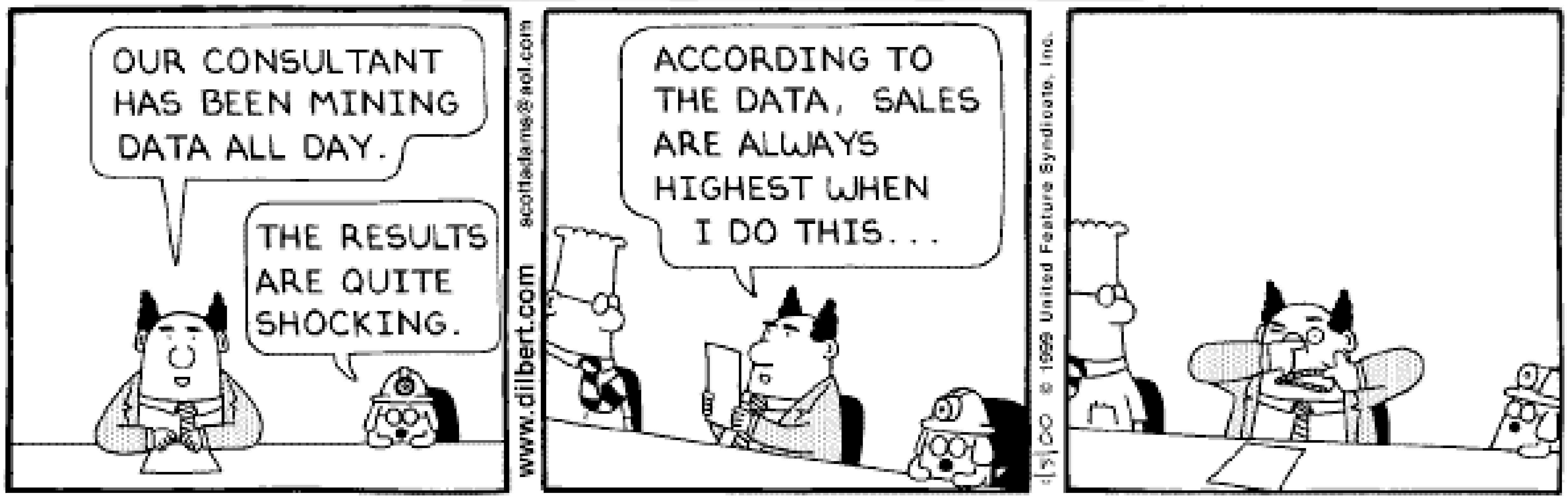
Frequent Pattern Mining



Universiteit Antwerpen

Bart Goethals
University of Antwerp, Belgium
ADReM research group
http://www.adrem.ua.ac.be/~goethals/

What this talk is about

- One of the most popular problems in computer science!
- [Agrawal, Imielinski, Swami, SIGMOD 1993]
  13th most cited across all computer science
  [Agrawal, Srikant, VLDB 1994]
  15th most cited across all computer science
- [Goethals, 2003]
  a nice survey
- several other very interesting papers

Pattern Mining

- Unsupervised learning
- Local (vs. global models)
- Useful for
  - large datasets
  - exploration: « what is this data like? »
  - building global models
- Less suitable for
  - well-studied and understood problem domains

Outline

- Mining association rules
- Algorithms
  - Apriori
  - Eclat
  - FP-growth
- Optimizations and Extensions
- Other pattern types
- General levelwise search
- Other interestingness measures

Back in 1993 ...

- Find associations between products

- For example: a supermarket

  - which products are frequently bought together?
  - do some products influence the sales of other products?
    e.g. “75% of people who buy beer, also buy chips”

Applications

- Supermarket
  - cross selling
  - product placement
  - special promotions
- Websearch
  - which keywords often occur together in webpages?
- Health care
  - frequent sets of symptoms for a disease
- Prediction
  - associative classifiers
- ...

Applications

- Basically works for all data that can be represented as a set of examples/objects having certain properties
  - patient / symptoms
  - movies / ratings
  - web pages / keywords
  - basket / products
  - …

Formally

- A **transaction database** is a collection of sets of items (transactions)
- An **itemset** is a set of items
- An **association rule** is an implication of the form X=>Y, with X and Y itemsets
- **Support Count** (SC) of an itemset X is the number of transactions that contain X
- **Support** of X (also **frequency** of X) = SC(X)/SC({})
- **Support** of an association rule X=>Y equals the support of X U Y
- **Confidence** of an association rule X=>Y = Support(X=>Y) / Support(X)

Problem

- Given:
  - a transaction database
  - a minimum support threshold
  - a minimum confidence threshold
- Find all rules X=>Y such that:
  - Support(X=>Y) > minsup
    (X=>Y is frequent)
  - Confidence(X=>Y) > minconf
    (X=>Y is confident)

Example

Tid	Transaction
1	shoes, socks, T-shirt
2	socks, sweater, pants
3	T-shirt, pants, socks
4	shoes, socks

- minimum support = 2
- minimum confidence = 2/3
- {shoes} ⇒ {socks} is a confident association rule with support = 0.5, confidence = 1
- {socks} ⇒ {shoes} is not
- Sweater can not appear in a rule

How?

- Solution #1:
  - Generate all possible rules
  - Count their supports and compute confidence
  - INTRACTABLE... (3^n possible combinations)
- Solution #2:
  - First, find all frequent itemsets
  - Second, split every frequent itemset Z in two parts X and Y, such that X ⇒Y is confident
    - Example: I = {A,B,C}
      test rules {A,B}⇒{C}, {AC}⇒{B}, {B,C}⇒{A},
      {A}⇒{B,C}, {B}⇒{A,C}, {C}⇒{A,B}

How to find all frequent itemsets?

- Solution #1:
  - Generate all possible itemsets
  - Count their support in DB
  - INTRACTABLE… (2^n possible combinations)

How to find all frequent itemsets?

- Solution #2:
  - Apriori
  - Rakesh Agrawal and Srikant Ramakrishnan [VLDB, 1994]
  - Heikki Mannila and Hannu Toivonen [KDD, 1994]

Apriori

- Key observation: (monotonicity)

  A subset of a frequent itemset must also be frequent, or,

  any superset of an infrequent itemset must also be infrequent!

Apriori

- An itemset is called a candidate itemset if all of its subsets are known to be frequent
- Solution:
  Iteratively find frequent itemsets with cardinality from 1 to k (k-itemset)

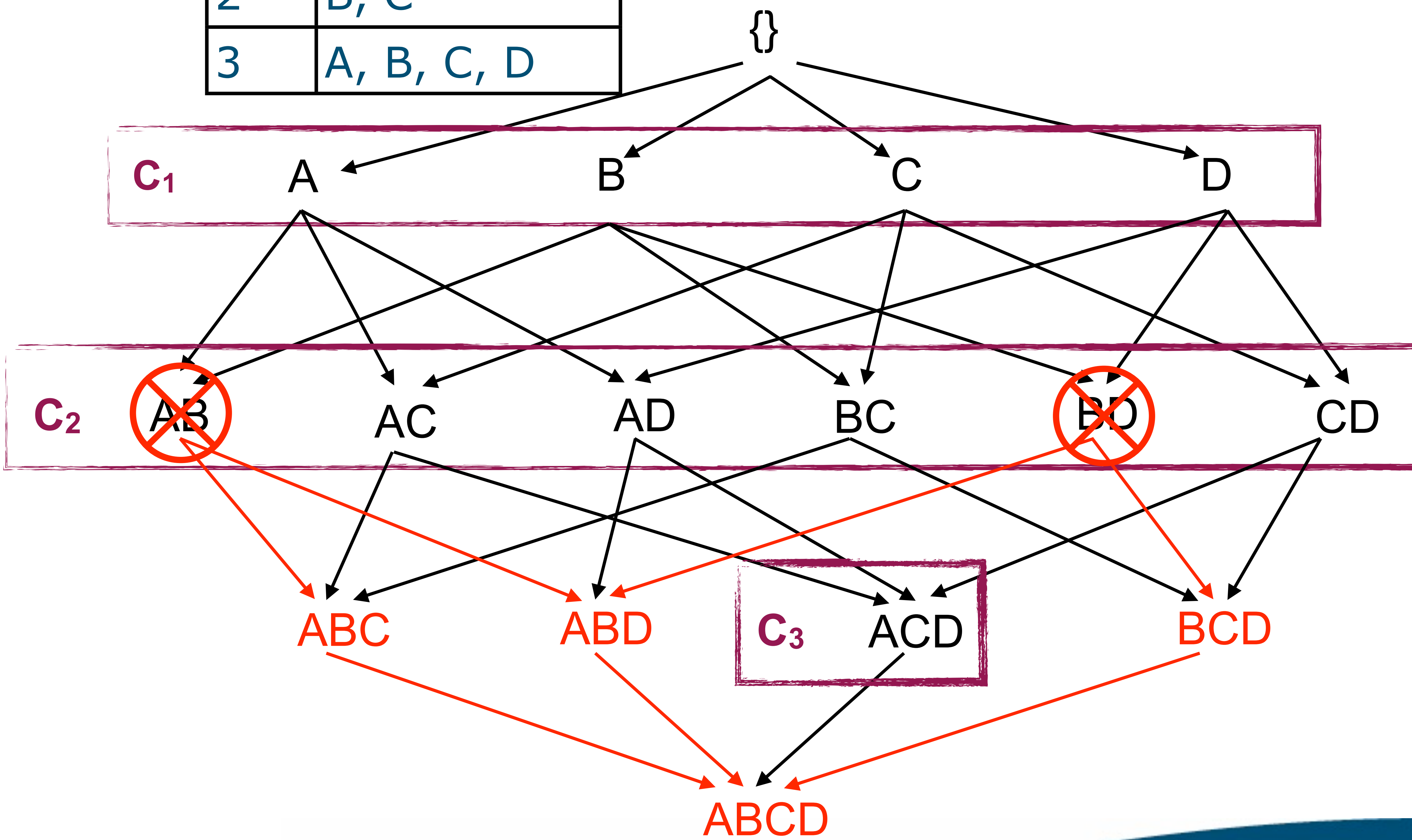
Example

- Start with small itemsets, only proceed with larger itemset if all subsets are frequent
- { A,B,C } is evaluated after {A}, {B}, {C}, {A,B}, {A,C}, and {B,C}, and only if all these sets are known to be frequent

Level-wise search

Tid	Items
1	A, C, D
2	B, C
3	A, B, C, D

minsup = 2



The Apriori Algorithm

C_k: candidate itemset of size k

L_k: frequent itemset of size k

L_1 = {frequent items};

for (k = 1; L_k != $\emptyset$; k++) **do begin**

    C_{k+1} = candidates generated from L_k;

    for each transaction t in database do

    increment the count of all candidates in C_{k+1} that are contained in t

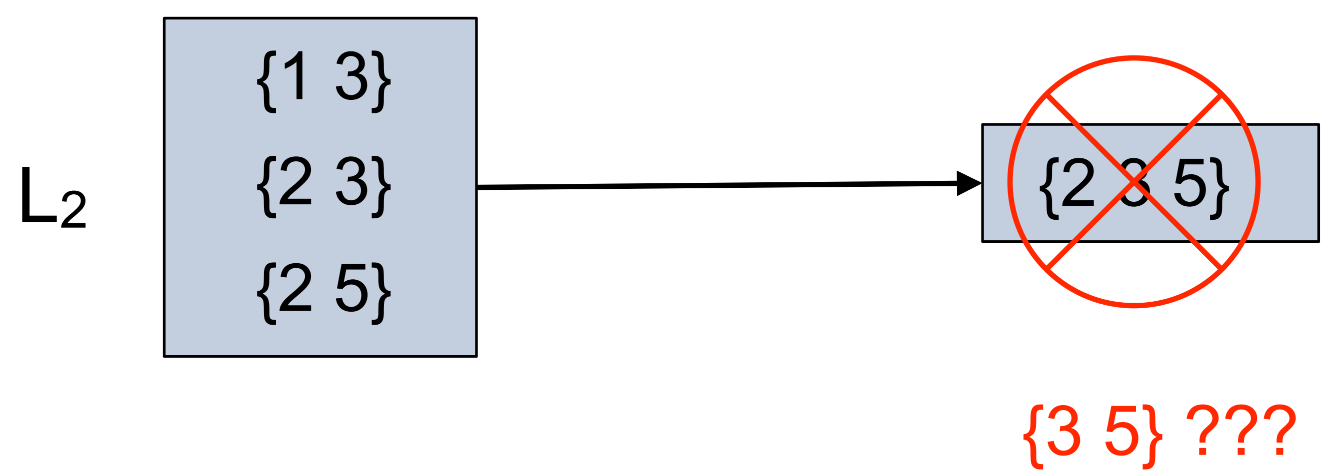
    L_{k+1} = candidates in C_{k+1} with min_support

    end

return $\cup_k L_k$;

Candidate Generation

- for all itemsets X, Y with X[:-1]=Y[:-1]
- X + Y[-1:] is a candidate itemset,
- only if all its subsets are known to be frequent
- note that {1,2,3} was not even considered



Example run

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

Scan D

C_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

L_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{5}	3

C_2

itemset
{1 2}
{1 3}
{1 5}
{2 3}
{2 5}
{3 5}

Scan D

C_2

itemset	sup
{1 2}	1
{1 3}	2
{1 5}	1
{2 3}	2
{2 5}	3
{3 5}	2

L_2

itemset	sup
{1 3}	2
{2 3}	2
{2 5}	3
{3 5}	2

C_3

itemset
{2 3 5}

Scan D

L_3

itemset	sup
{2 3 5}	2

Apriori's main problem

- In every count step we have to do a very costly scan over the complete database.

Optimizations

- Dynamic Itemset Counting [Brin et al., 1997]
  - interrupt algorithm after every M transactions and already generate larger candidates if possible
- Partition [Savasere et al., 1995]
  - partition database, and mine each part separately (using relative minsup!)
  - Union of all frequent itemsets of all parts are a superset of all frequent itemsets in complete database!
  - Extra pruning step
- Sampling [Toivonen, 1995]
  - Run apriori on small sample of DB
  - Correct result

Current Research

- Until today, many researchers still try to find new techniques, and improve Apriori
  - Optimized for sparse/dense data
  - Optimized for many/few items
- Implementation issues are important
  - How to implement the counting step
  - How to read the database
  - How to generate the candidates
  - How to prune the candidates
  - Ordering of items is important!
- For more info: visit http://fimi.cs.helsinki.fi/

What if DB fits in memory?

- Faster counting of supports!
- Two new techniques differ in counting strategy and how the database is represented in memory
  - Eclat [Zaki et al., KDD 1997]
  - FP-growth [Han et al., SIGMOD 2000]

Eclat: tidlist

- For every item, a list of transaction id's is stored in which the item occurs, denoted by tidlist

- For every itemset, its tidlist equals the intersection of the tidlists of two of its subsets

Eclat: tidlist example

{a}	1	2	3	4	5
{b}	1	3	5	6	7
{a,b}	1	3	5		

1	{a,b}
2	{a}
3	{a,b}
4	{a}
5	{a,b}
6	{b}
7	{b}

Eclat: algorithm

- In principle Apriori could be used together with intersection based support counting
- Memory usage, however, would blowup!
- Therefore, a depth-first approach is used

Divide and conquer

1. Find all itemsets containing {a}
2. Find all itemsets not containing {a}

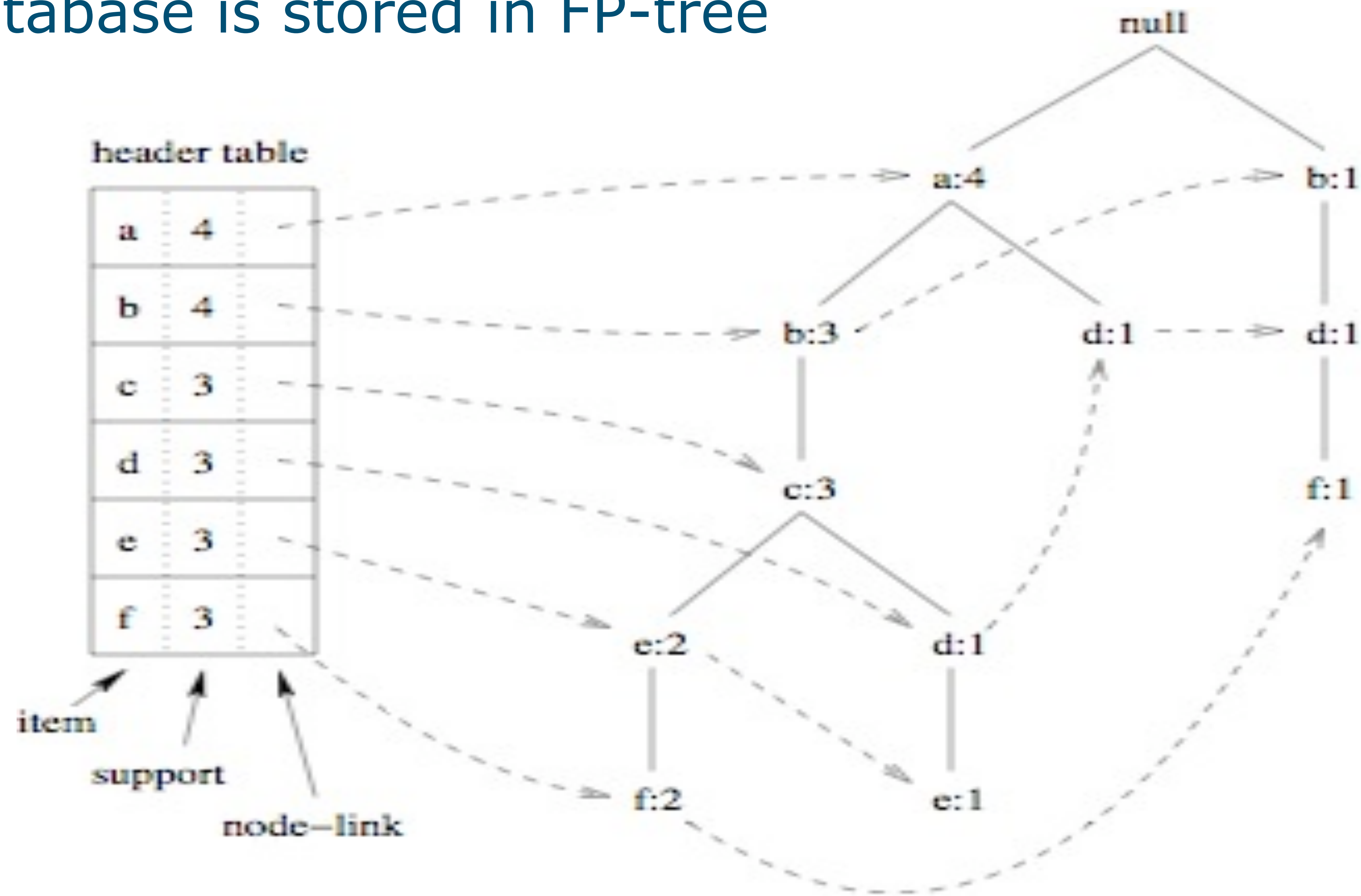
- For 1. Only transactions containing {a} are necessary ({a} can be removed)
  => *{a}-conditional database*
- For 2. {a} can be removed from all transactions
- Apply recursively

Eclat: algorithm

1. Get tidlist for each item (DB scan)
2. Tidlist of {a} is exactly the list of transactions containing {a}
3. Intersect tidlist of {a} with the tidlists of all other items, resulting in tidlists of {a,b}, {a,c}, {a,d}, …
   = {a}-conditional database (if {a} removed)
4. Repeat from 1 on {a}-conditional database
5. Repeat for all other items

FP-growth

- Database is stored in FP-tree



FP-growth

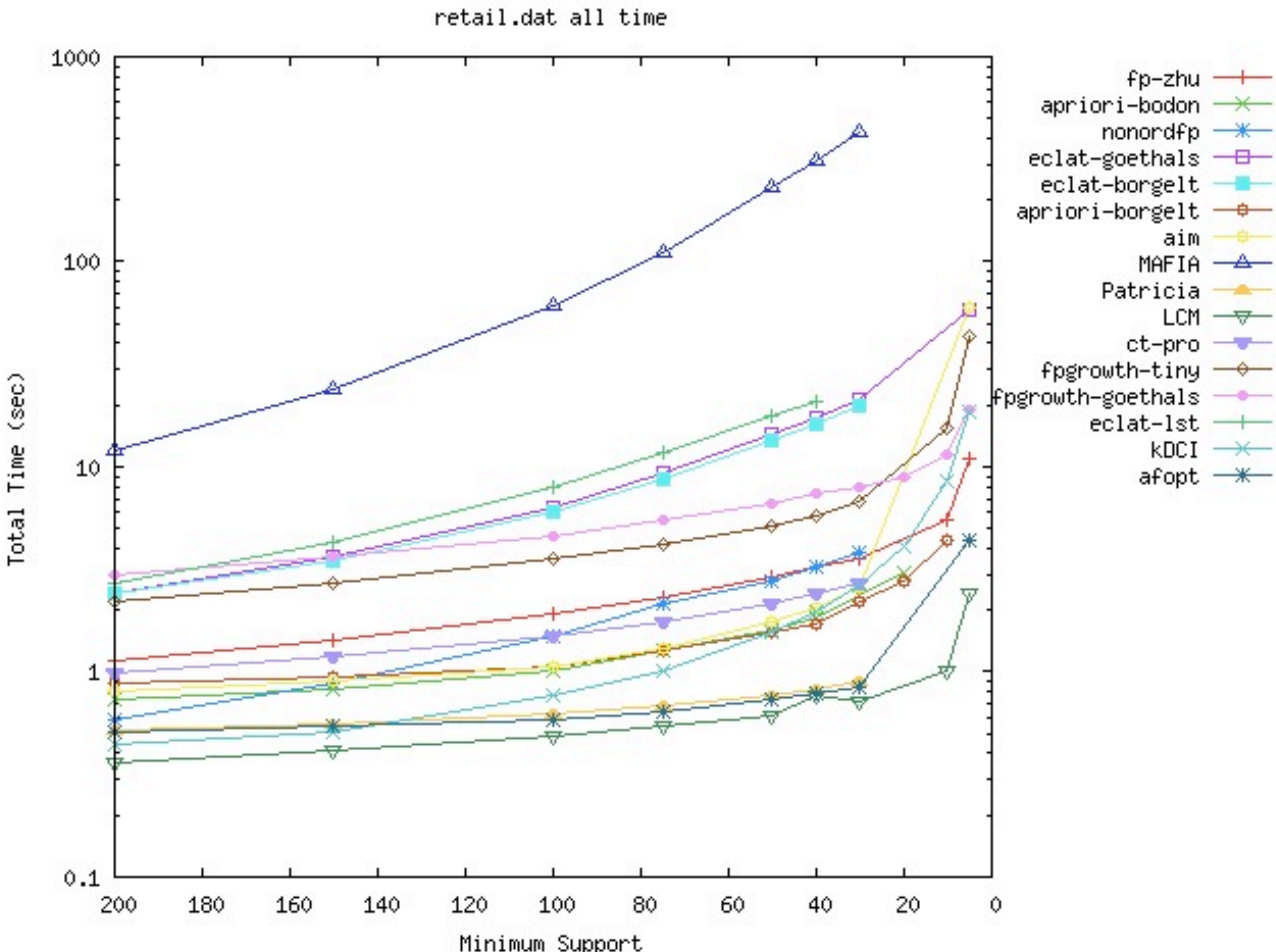
- Divide and conquer strategy is used
  1. Find all itemsets containing {a}
  2. Find all itemsets not containing {a}
- For 1. Only transactions containing {a} are necessary ({a} can be removed)
  => *{a}-conditional database*
- For 2. {a} can be removed from all transactions
- Apply recursively

Apriori vs. Eclat vs. FP-growth

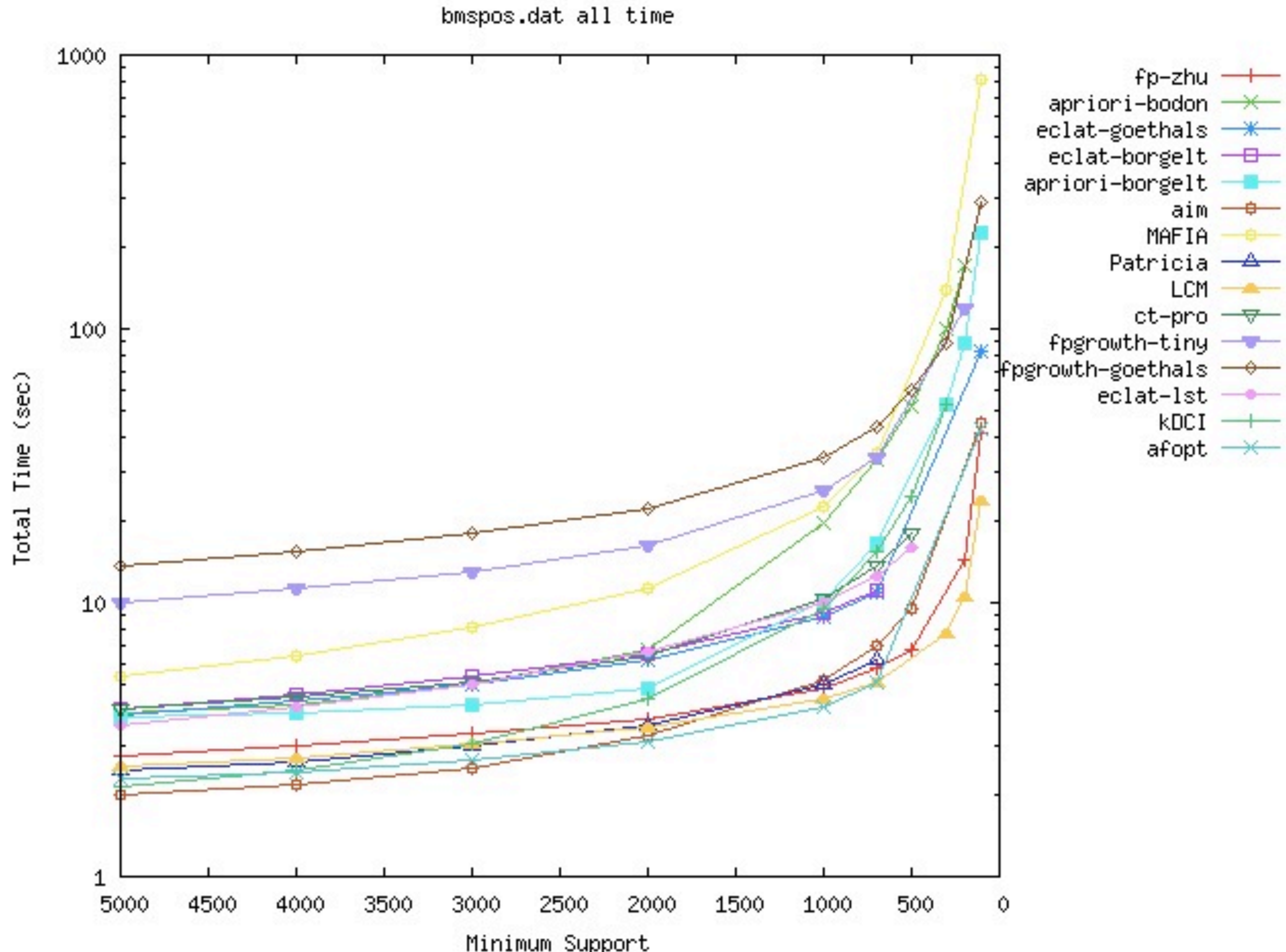
- Which is best? Depends on data
- Apriori better for huge databases
- Eclat most of the time better than FP-growth
- Many optimizations exist! (see FIMI)

- FP-growth paper title says: “Mining Frequent Patterns without candidate generation”
- Where did the candidates go?

Some FIMI results



retail.dat all time
Total Time (sec)
Minimum Support
1000
100
10
1
0.1
200
180
160
140
120
100
80
60
40
20
0
fp-zhu
apriori-bodon
nonordfp
eclat-goethals
eclat-borgelt
apriori-borgelt
aim
MAFIA
Patricia
LCM
ct-pro
fpgrowth-tiny
fpgrowth-goethals
eclat-lst
kDCI
afopt


bmspos.dat all time
Total Time (sec)
1000
100
10
1
5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
Minimum Support
fp-zhu
apriori-bodon
eclat-goethals
eclat-borgelt
apriori-borgelt
aim
MAFIA
Patricia
LCM
ct-pro
fpgrowth-tiny
fpgrowth-goethals
eclat-lst
kDCI
afopt

Some FIMI conclusions

- There is no clear winner
- Much depends on implementation details
- Experiments should be reproducible and therefore source code should be available!

Extensions

- Maximal Itemset Mining [Bayardo, 1998]
  - One might not be interested in all frequent itemsets, but only in the maximal ones
  - optimized algorithms exist
- Closed Itemset Mining [Pasquier et al., 1999]
  - Suppose A=>X holds with 100% confidence
  - Then, every itemset containing A also occurs with all subsets of X, with exactly the same support
  - Only reporting A U X is sufficient

Extensions

- Non derivable Itemset Mining [Calders et al, 2002]
  - support bounds of an itemset can be derived from its subsets using the inclusion-exclusion principle
  - if these bounds are tight, then the support of that itemset is derivable
  - only reporting the non-derivable itemsets is sufficient

Outline

- Mining association rules
- Algorithms
  - Apriori
  - Eclat
  - FP-growth
- Optimizations and Extensions
- Other pattern types
- General levelwise search
- Other interestingness measures

Complex Patterns

- Sets

- Sequences
- Graphs
- Relational Structures

- Generation and Counting of such patterns becomes much more complex too!

Sequences

- CGATGGGCCAGTCGATACGTCGATGCCGATGTCACGA

Patterns in Sequences

- Substrings
- Regular expressions (bb|[^b]{2})
- Partial orders
- Directed Acyclic Graphs
- Episodes

Episode mining

- Given a sequence of events
- ABCDBABDABDBSBDBCSBABCBSBCA
- A sequential episode is an ordered list of events
- Goal: Find all frequently occurring (sequential) episodes

Episode Mining

- Event sequence: sequence of pairs (e,t), e is an event, t an integer indicating the time of occurrence of e.
- An linear episode is a sequence of events $\langle e_1, \ldots, e_n \rangle$.
- A window of length w is an interval [s,e] with $(e-s+1) = w$.
- An episode $E=\langle e_1, \ldots, e_n \rangle$ occurs in sequence $S=\langle (s_1,t_1), \ldots, (s_m,t_m) \rangle$ within window W=[s,e] if there exist integers $s \leq i_1 < \ldots < i_n \leq e$ such that for all $j=1 \ldots n$, (e_j,i_j) is in S.

- The w-support of an episode $E=<e_1, \ldots, e_n>$ in a sequence $S=<(s_1,t_1), \ldots, (s_m,t_m)>$ is the number of windows W of length w such that E occurs in S within window W.
- Note: If an episode occurs in a very short time span, it will be in many subsequent windows, and thus contribute a lot to the support count!
- An episode $E_1=<e_1, \ldots, e_n>$ is a sub-episode of $E_2=<f_1,\ldots,f_m>$, denoted $E_1 \leq E_2$ if there exist integers $1 \leq i_1 < \ldots < i_n \leq m$ such that for all $j=1\ldots n$, $e_j=f_{i_j}$.

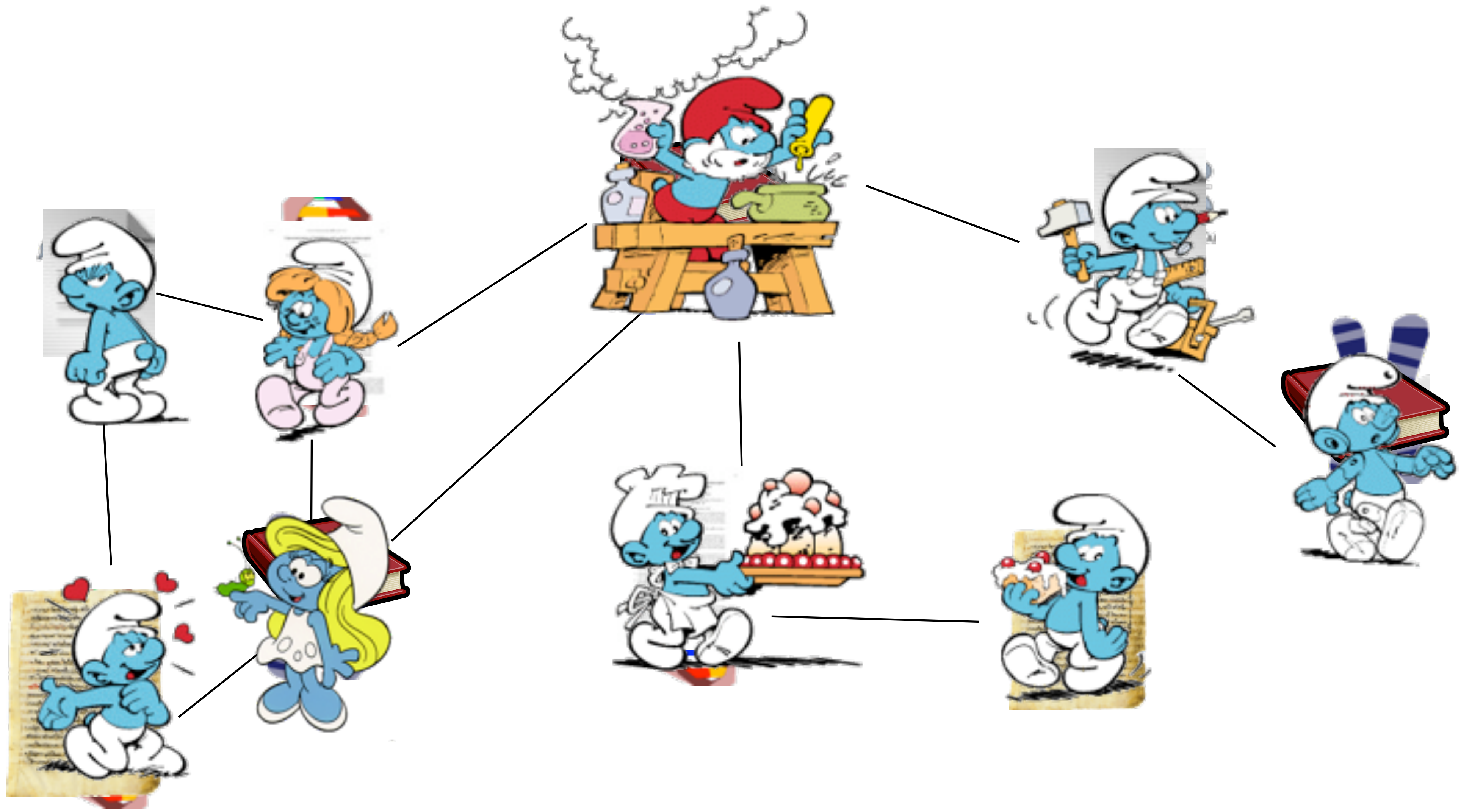
Example

- S = <(b,1), (a,2), (a,3), (c,4), (b,5), (a,6), (a,7), (b,8), (c,9) >
- E = < b, a, c >
- E occurs in S within window [0,4], within [1,4], within [5,9], …
- The 5-support of E in S is 3, since E is only in the following windows of length 5: [0,4], [1,5], [5,9]
- < b, a, a, c > is a sub-episode of <a, b, c, a, a, b, c>.

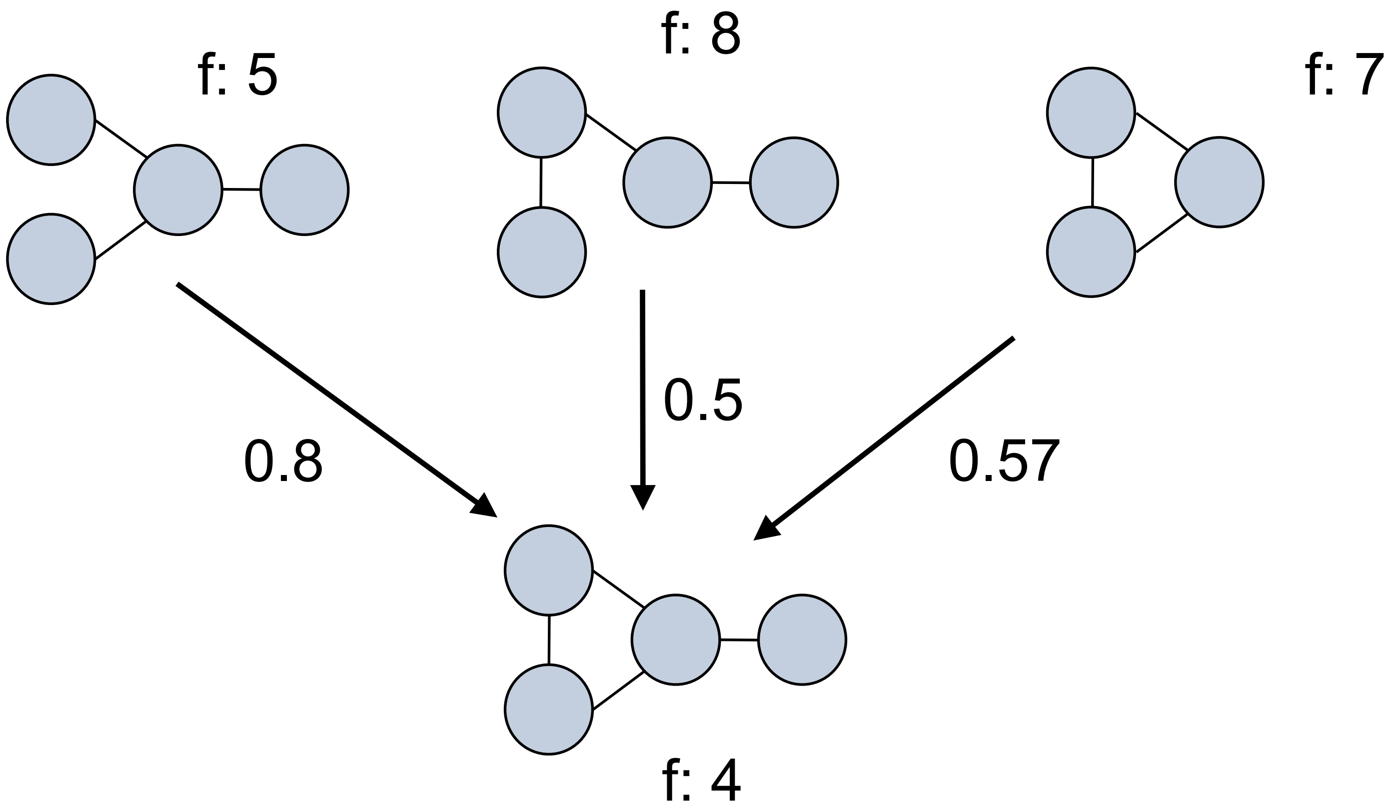
Problem

- Given a sequence w, a minimal support minsup, and a window width w, find all episodes that have a w-support above the minimum support.
- Monotonicity
  Let S be a sequence, E_1, E_2 episodes, w an integer.
  If $E_1 \leq E_2$, then the w-freq(E_2) $\leq$ w-freq(E_1).
- We can again apply a level-wise algorithm like Apriori.
- Start with small episodes, only proceed with a larger episode if all sub-episodes are frequent.
- <a,a,b> is evaluated after <a>, <b>, <a,a>, <a,b>, and only if all these episodes were frequent.

Graphs



Patterns and Rules over Graphs



Relational Databases

$Likes(Drinker, Beer)$
$Visits(Drinker, Bar)$
$Serves(Bar, Beer)$

Likes

Drinker	*Beer*
Allen	Duvel
Allen	Trappist
Carol	Duvel
Bill	Duvel
Bill	Trappist
Bill	Jupiler

Visits

Drinker	*Bar*
Allen	Cheers
Allen	California
Carol	Cheers
Carol	California
Carol	Old Dutch
Bill	Cheers

Serves

Bar	*Beer*
Cheers	Duvel
Cheers	Trappist
Cheers	Jupiler
California	Duvel
California	Jupiler
Old Dutch	Trappist

Patterns in RDBs

- Query 1:
  - Select L.drinker, V.bar
    From Likes L, Visits V
    Where V.drinker = L.drinker
    And L.beer = 'Duvel'
- Query 2:
  - Select L.drinker, V.bar
    From Likes L, Visits V, Serves S
    Where V.drinker = L.drinker
    And L.beer = 'Duvel'
    And S.bar = V.bar
    And S.beer = 'Duvel'

Patterns in RDBs

- Association Rule:

  Query 1 => Query 2

- If a person that likes Duvel visits bar, then that bar serves Duvel

Pattern Mining in general

- Given:
  - A database
  - A partially ordered class of patterns
  - An interestingness measure (e.g. support) which is monotone w.r.t. partial order
- Problem:
  - Find all interesting patterns

Solution

- Generate ‘small’ set of candidate patterns
- Test interestingness measure
- Remove all uninteresting patterns from search space according to monotonicity
- Repeat until all interesting patterns have been found

- [Mannila et al., DMKD 1(3), 1997]

Other constraints or interestingness

- When monotone, Apriori technique can be used
- What if they are not monotone?
- For example:
  - minimum size of itemset or total price of itemset
  - database can be reduced!
- Another example:
  - Mining Tiles

100100101101001011010101001000101001101010100100101101010100101010
0010000001100000011000100011001101000100010001000000110001000110 1001
110000000100000001010011111011110100101001110000000101001111100010
100101100101011001010101000100010010101010100101100101010100011100
1000100000100100000100010001100100010000101010001000010001000110 1001
010011100100111001010011111011010000100111010011100101001111101000
001001010100010101010101001100000100100101001001010101010100110100
101101010101010101001001010101101010100100101101010100100101010010
001100010001000100010000010001100010001000000110001000100000100010 0
000101001111010011100000000100101001110000000101001110000000010000
100101010101010101001001010001101010100101100101010100100101000100
000100010001000100010000010001001010100010000100010001000001001010
100101001111010011100000000100100111010011100101001110000000010100
110101010011010100100101101011111111111010011010101001001011010101 0
010001000100010001000001100011111111110100010001000100000110001000
010100111001001110000000010101111111111101001010011100000000101001 11
010101010011010100101100101011111111110100101010101001011001010101 0
000101010001010100010000100011111111100010000101010001000010001000
010011101000111010011100101000000000010100001001110100111001010011 1
010010100100101001001010101011111111100010010010100100101010101010
010010010110100101101010100100000000110101010010010110101010010010
000100000011100000011000100010111111110100010001000000110001000100 000
101001101010100100101101010100101010110100111000000010100111000000
101000100010001000000110001000110100111010101001011001010101001001 0
110100101001110000000101001111100010100101010100010000100010001000 00
010010101010100101100101010100011100010011101001110010100111000000
000100000101010001000010001000110100110101010010010110101010010010 1
010000100111010011100101001111101000100010001000000110001000100000 1
000100100101001001010101010100110100101001110000000101001110000000
101010100100101101010100100101010010101010100101100101010100100101
100010001000000110001000100000010001000010101000100001000100010000 01
101001110000000101001110000000010000100111010011100101001110000000
101010100101100101010100100101000100100101001001010101010101000101
001010100010000100010001000001001010100100010001010010101001010110
100111010011100101001110000000010100010001000010000100001101010110

Motivation

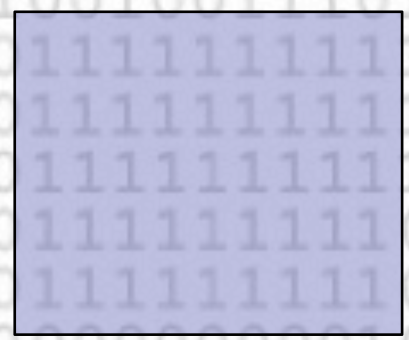
- What makes my database unique?
- Describe my database using only a small description
- For example: using itemsets

Motivation

- Which itemsets describe my database best?

- Interestingness measures?
- Most are subjective depending on the specific application
- Support/Frequency is objective

Tiles



- A *tile* is an itemset together with the transactions in which it occurs

Tiles

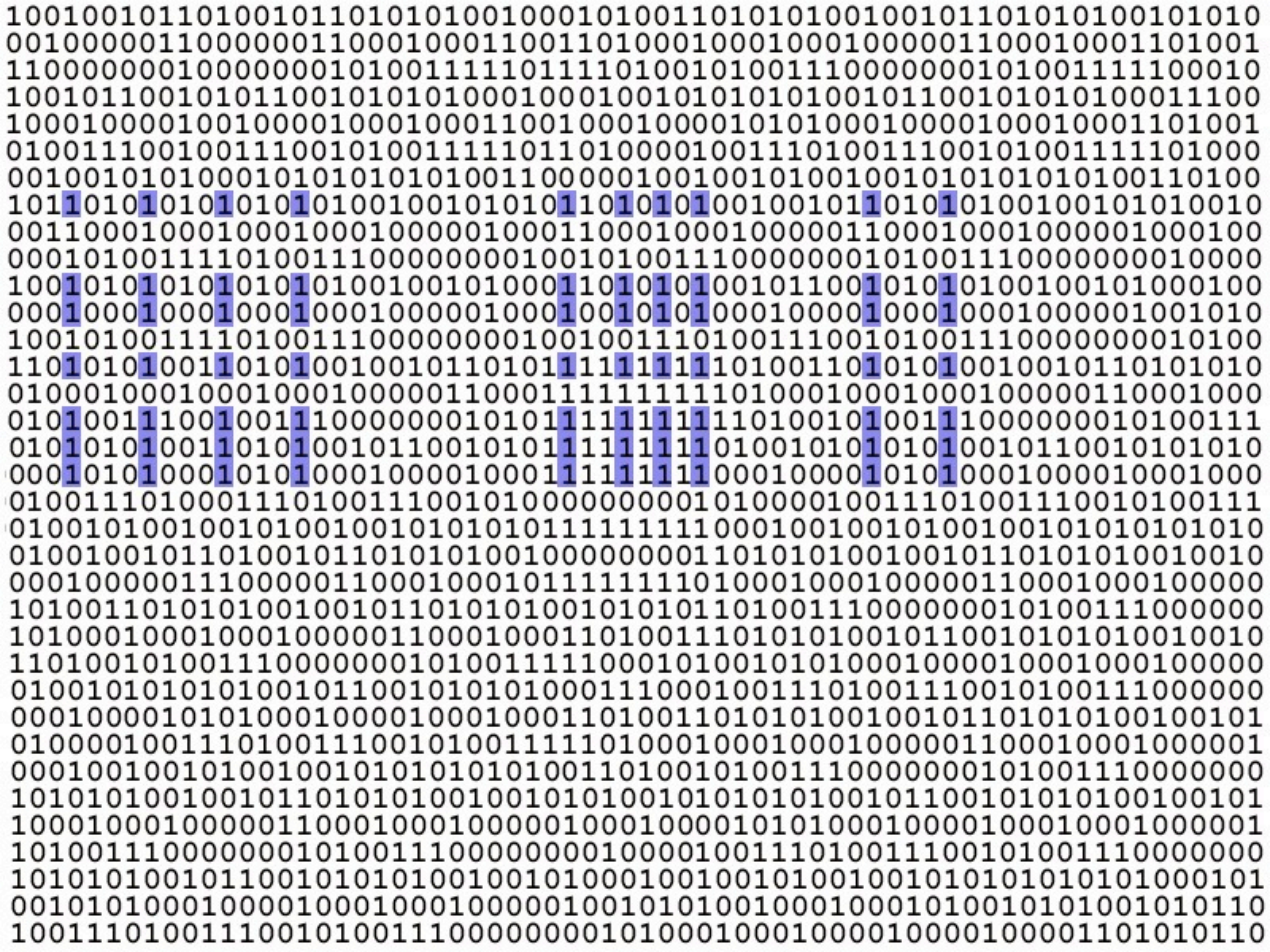
- We only consider maximal tiles (= closed)

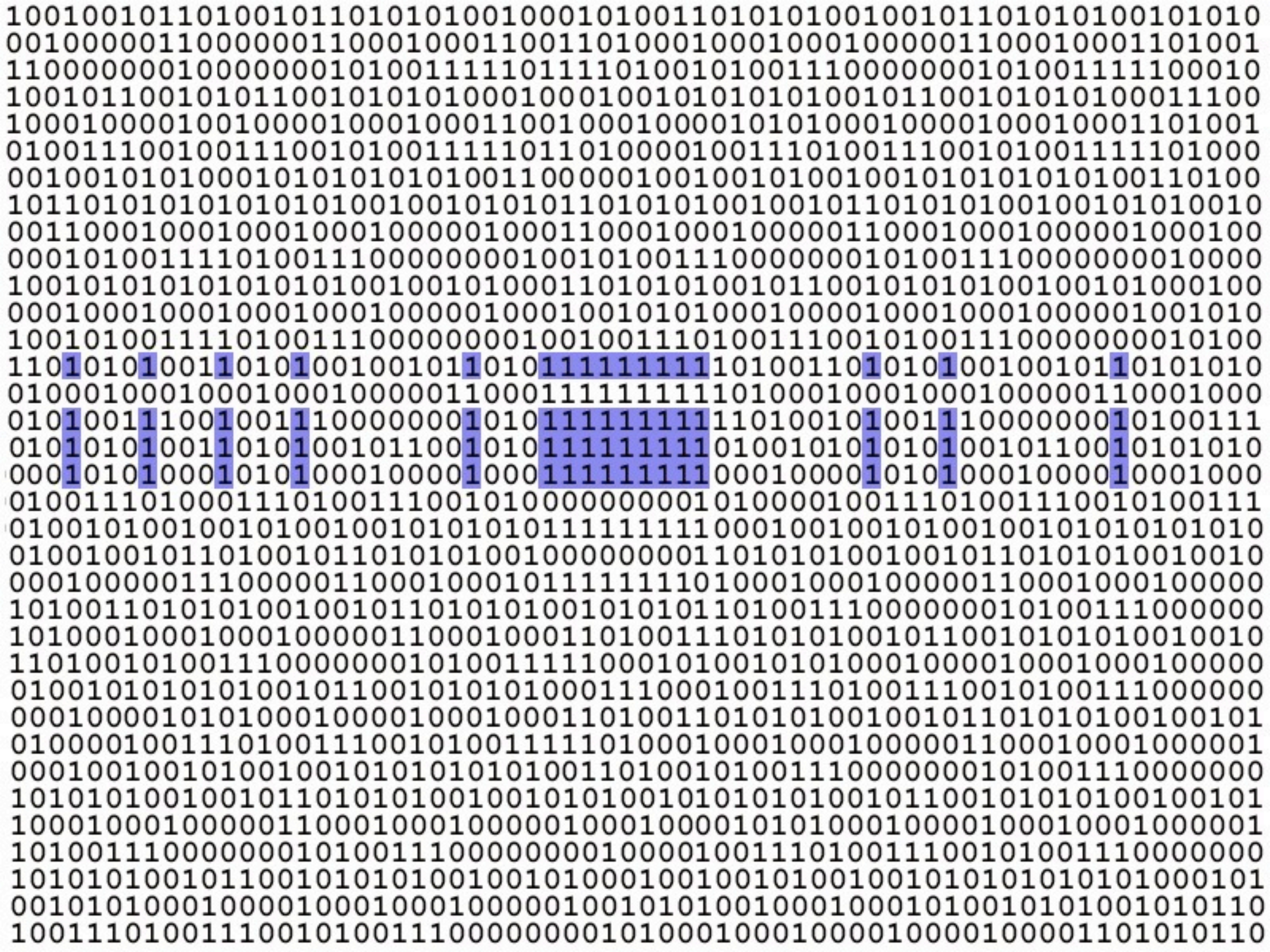
Tile Mining

- The area of a tile is the number of 1's occurring in it
- Goal: Find all tiles with area at least s

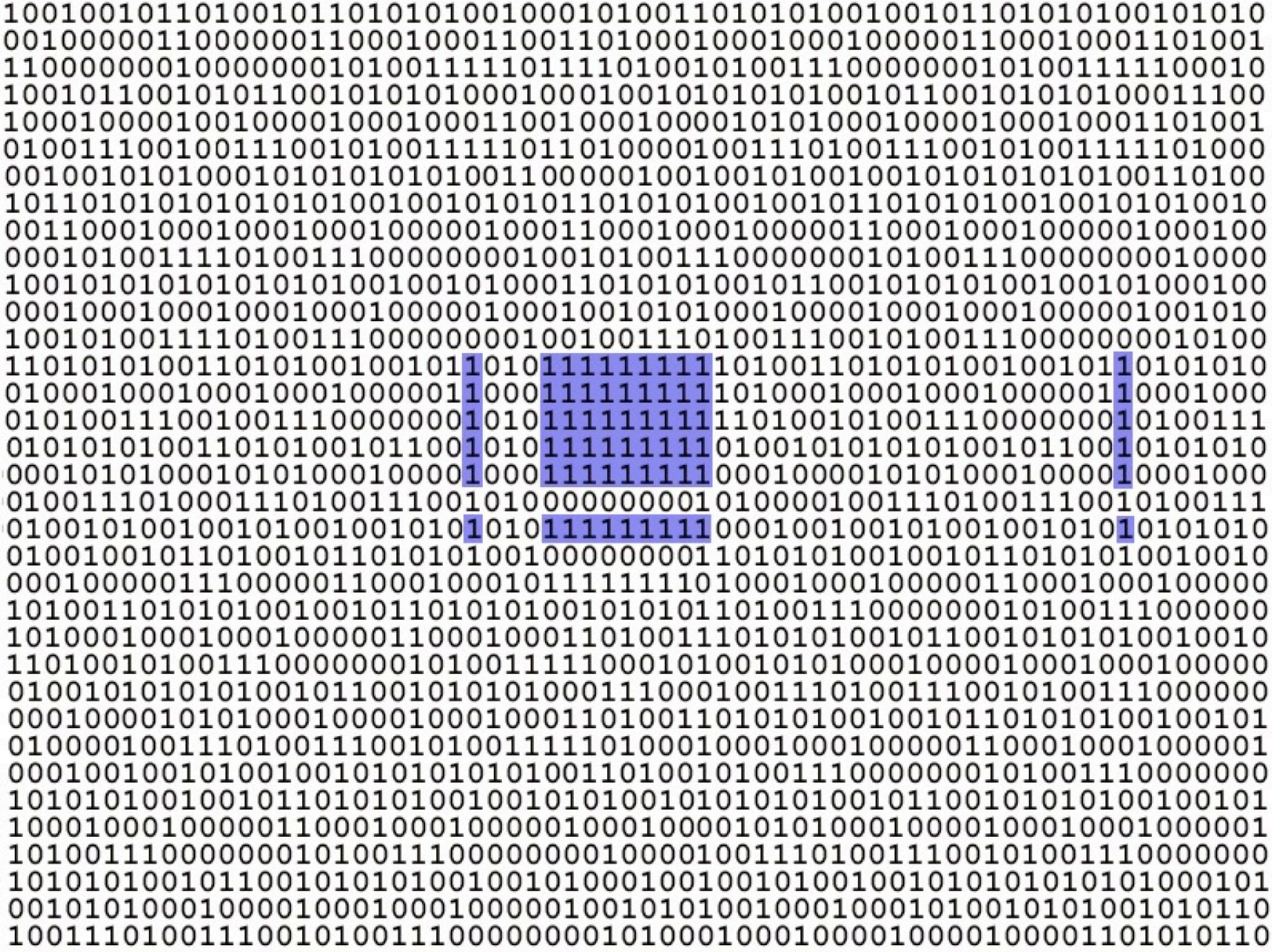
1001001011010010110101010010001010011010101001001011010101010100101010
0010000011000000110001000110011010001000100010000001100010001101001
1100000001000000010100111110111101001010011100000000101001111100010
1001011001010110010101010100010001001010101010100101100101010100011100
1000100001001000010001000110010001000010101000100001000100011101001
0100111001001110010100111110110100001001110100111001010011111101000
0010010101000101010101010100110000010010010100100101010101010100110100
1011010101010101010100100101010110101010100100101101010100100101010010
0011000100010001000100000100011000100010000011000100010000001000100
000101001111010011100000000100101001110000000010100111000000000010000
1001010101010101010100100101000110101010100101100101010101001001010000100
0001000100010001000100000100010010101000100001000100010000001001010
1001010011110100111000000000100100111010011100101001110000000000010100
1101010100110101001001011010111111111101001101010100100101101010101010
0100010001000100010000011000111111111101000100010001000000110001000
0101001110010011100000000101011111111110100101001110000000010100111
0101010100110101001011001010111111111101001010101010010110010101010
0001010100010101000100001000111111111100010000101010001000010001000
0100111010001110100111001010000000000101000010011101001110010100111
0100101001001010010010101010111111111100010010010100100101010101010
0100100101101001011010101001000000000110101010010010110101010100010010
0001000000111000000110001000101111111110100010001000000110001000100000
1010011010101001001011010101001010101101001110000000101001110000000
1010001000100010000011000100011010011101010100101100101010100100100010
1101001010011100000000101001111100010100101010100010000100010001000100000
0100101010101001011001010101000111000100111010011100101001110000000
0001000001010100010000100010001101001101010100100101101010100100101
0100001001110100111001010011111010001000100010000001100010001000001
0001001001010010010101010101001101001010011100000000101001110000000
1010101001001011010101001001010100101010101001011001010101001001001001
1000100010000011000100010000010001000010101000100001000100010001000001
1010011100000000101001110000000001000010011101001110010100111000000
1010101001011001010101001001010001001001010010010101010101010101000101
0010101000100001000100010000001001010100100010001010010101010010101110
1001110100111001010011100000000010100010001000010000100000110101010110

1001001011010010110101010010001010011010101001001011010101001010 10
0010000001100000011000100011001101000100010001000000110001000110 1001
1100000001000000001010011111011110100101001110000000101001111100010
1001011001010110010101010001000100101010101001011001010101000111 00
1000100001001000010001000110010001000010101000100001000100011 01001
0100111001001110010100111110110100001001110100111001010011111101000
0010010101000101010101010011000001001001010010010101010101001101 00
1011010101010101010100100101010110101010010010110101010010010101 0010
0011000100010001000100000100011000100010000001100010001000000100 0100
0001010011110100111000000000100101001110000000101001110000000001 0000
1001010101010101010100100101000110101010010110010101010010010100 0100
0001000100010001000100000100010010101000100001000100010000001001 010
1001010011110100111000000000100100111010011100101001110000000001 0100
1101010100110101010010010110101111111110100110101010010010110101 010
0100010001000100010000001100011111111110100010001000100000011000 1000
0101001110010011100000000101011111111111010010100111000000001010 0111
0101010100110101010010110010101111111110100101010101010010110010101 010
0001010100010101000100001000111111111000100001010100010000100010 00
0100111010001110100111001010000000001010000100111010011100101001 11
0100101001001010010010101010101111111110001001001010010010101010101 010
0100100101101001011010101001000000001101010100100101101010100100 10
0001000000111000000110001000101111111101000100010000001100010001 00000
1010011010101010010010110101010010101011010011100000000101001110 00000
1010001000100010000011000100011010011101010100101100101010100100 10
1101001010011100000000101001111110001010010101000100001000100010 0000
0100101010101001011001010101000111000100111010011100101001110000 00
0001000010101000100001000100011010011010101001001011010101010010 0101
0100001001110100111001010011111010001000100010000001100010001000 001
0001001001010010010101010101001101001010011100000000101001110000 000
1010101001001011010101010010010101001010101010010110010101010010 0101
1000100010000001100010001000000100010000101010001000010001000100 0001
1010011100000000101001110000000010000100111010011100101001110000 000
1010101001011001010101001001010001001001010010010101010101010100 0101
0010101000100001000100010000001001010100100010001010010101010010 10110
1001110100111001010011100000000010100010001000010000100001101010 10110





```
1001001011010010110101010010001010011010101001001011010101001010l0
0010000001100000011000100011001101000100010001000000110001000110l001
1100000001000000001010011111011110100101001110000000101001111100010
1001011001010110010101010001000100101010101001011001010101000111100
1000100001001000010001000110010001000010101000100001000100011010 01
0100111001001110010100111110110100001001110100111001010011111101000
0010010101000101010101010011000001001001010010010101010101001101 00
1011010101010101010100100101010111010101001001011010101001001010100 10
001100010001000100010000010001100010001000000110001000100000010001 00
0001010011110100111000000000100101001110000000101001110000000001 0000
100101010101010101010010010100011010101001011001010101001001010001 00
00010001000100010001000000100010010101000100001000100010000001001010
1001010011110100111000000000100100111010011100101001110000000001010 0
110101010011010101001001011010111111111101001101010100100101101010 10
0100010001000100010000001100011111111110100010001000100000011000100 0
0101001110010011100000000101011111111111010010101001110000000010100111
0101010100110101001011001010111111111101001010101010100101100101010 10
0001010100010101000100001000111111111100010000101010001000010001000
0100111010001110100111001010000000001010000100111010011100101001 11
0100101001001010010010101010111111111000100100101001001010101010 10
010010010110100101101010100100000000110101010010010110101010010010
000100000011100000011000100010111111110100010001000000110001000100000
101001101010100100101101010100101010110100111000000010100111000000
1010001000100010000011000100011010011101010100101100101010100100 10
1101001010011100000000101001111100010100101010001000010001000100000
0100101010101001011001010101000111000100111010011100101001110000 00
000100000101010001000010001000110100110101010010010110101010010 0101
0100001001110100111001010011111010001000100010000001100010001000 001
000100100101001001010101010100110100101001110000000101001110000 000
101010100100101101010100100101010010101010100101100101010100100 101
1000100010000001100010001000000100010000101010001000010001000100000 1
101001110000000010100111000000000100001001110100111001010011100000 00
1010101001011001010101001001010001001001010010010101010101010001 01
001010100010000100010001000000100101010010001000101001010100101 0110
10011101001110010100111000000000101000100010000100001000011010 10110
```



1001001011010010110101010010001010011010101001001011010101001010
0010000001100000011000100011001101000100010001000000110001000110100 1
1100000001000000010100111110111101001010011100000000101001111100010
1001011001010110010101010001000100101010101001011001010101000111 00
1000100001001000010001000110010001000010101000100001000100011010 01
0100111001001110010100111110110100001001110100111001010011111010 00
0010010101000101010101010011000001001001010010010101010101001101 00
1011010101010101010010010101011010101001001011010101001001010100 10
0011000100010001000100000100011000100010000011000100010000010001 00
0001010011110100111000000000100101001110000000010100111000000000 10000
1001010101010101010100100101000110101010010110010101010010010100 0100
0001000100010001000100000100010010101000100001000100010000010010 10
1001010011110100111000000000100100111010011100101001110000000001 0100
1101010100110101001001011010111111111101001101010100100101101010 10
0100010001000100010000011000111111111101000100010001000000110001000
0101001110010011100000000101011111111111010010100111000000001010011 1
0101010100110101001011001010111111111101001010101010010110010101010
0001010100010101000100001000111111111000100001010100010000100010 00
0100111010001110100111001010000000001010000100111010011100101001 11
0100101001001010010010101010111111111000100100101001001010101010 10
0100100101101001011010101001000000001101010100100101101010100100 10
0001000000111000000110001000101111111101000100010000001100010001 00000
1010011010101001001011010101001010101101001110000000101001110000 00
1010001000100010000011000100011010011101010100101100101010100100 10
1101001010011100000000101001111100010100101010001000010001000100 000
0100101010101001011001010101000111000100111010011100101001110000 00
0001000010101000100001000100011010011010101001001011010101001001 01
0100001001110100111001010011111010001000100010000011000100010000 01
0001001001010010010101010101001101001010011100000000101001110000 000
1010101001001011010101001001010100101010101001011001010101001001 01
1000100010000011000100010000010001000010101000100001000100010000 01
1010011100000000101001110000000001000010011101001110010100111000 0000
1010101001011001010101001001010001001001010010010101010101010001 01
0010101000100001000100010000010010101001000100010100101010010101 10
1001110100111001010011100000000010100010001000010000100001101010 110

Can we efficiently find them?

- Area of tiles is not monotone w.r.t. set inclusion ☹
- Mining tiles and tilings is NP-hard ☹ (~maximum edge biclique problem)

The LTM algorithm

- Branch and bound
- Traverse itemset lattice depth-first (like Eclat and FP-growth)

- At every node, bound the size of the largest tile that can still be found

The bound

- For every item, we count the number of transactions of size larger than k in which the item occurs

1	100
2	80
3	60
4	40
5	20
…	…

100

160

180

160

100

…

The Dynamics

- If an item can not occur in a large tile anymore, we can remove it
- If a transaction can not contribute to a large tile anymore, we can remove it
- If an item in a specific transaction can not contribute to a large tile, we can remove it from that transaction
- Results in shorter transactions
- Recompute the bounds

The End

C++ Implementations of Apriori, Eclat, FP-growth and several other algorithms are available on my webpage

http://www.adrem.ua.ac.be/~goethals/software/

and on

http://fimi.cs.helsinki.fi/

Sources: I used some material from slides of Jiawei Han and Toon Calders