MAFIA: A Maximal Frequent Itemset Algorithm for Transactional Databases

Authors: Doug Burdick, Manuel Calimlim, Johannes Gehrke
Presented by: Benjamin Chu
CMPUT 695 Fall 2004

Outline

- Introduction
- Related Work
- Algorithmic Components
- Database Representation
- Experimental Results
- Comparison - DepthProject
- Conclusion

Introduction - Problem

- Association Rule Mining, Two Phases
  1) Find Frequent Itemsets (**FI**)
  2) Generate "interesting" patterns
- Most time in association rule mining is spent in finding the frequent itemsets
- When itemsets are long (g.t. 15-20 items), finding entire FI can be infeasible.

Introduction - Solutions

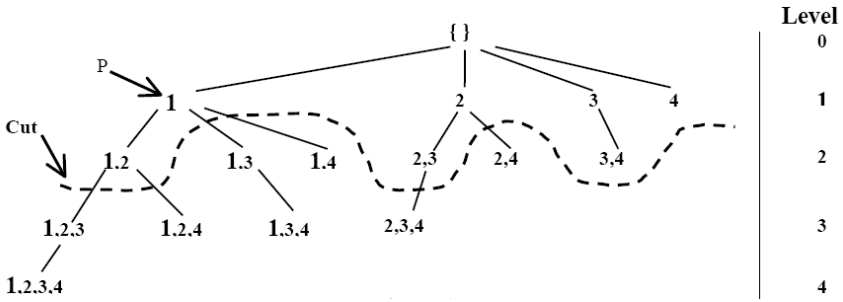
- Alternatives to "FI"
  - Frequent Closed Itemsets (**FCI**)
    - FCI: Itemset X is *closed* if there are no supersets with the same support.
  - Maximal Frequent Itemsets (**MFI**)
    - MFI: Itemset X is *maximally* frequent if no superset of X is frequent.

$$\mathbf{MFI} \subseteq \mathbf{FCI} \subseteq \mathbf{FI}$$

Introduction - MAFIA

- Integrates new and old ideas into practical algorithm for solving MFI problem
- Problem of mining frequent itemsets viewed as finding a *cut* through itemset lattice
- All items above cut are frequent itemsets
- All items below cut are infrequent itemsets

Introduction - Item Subset Lattice / Tree



- Head(N) – itemset identifying node N
- Tail(N) – set of all possible extensions of the node N
- HUT – Head U(nion) Tail

Outline

- Introduction
- Related Work
- Algorithmic Components
- Database Representation
- Experimental Results
- Comparison – DepthProject
- Conclusion

Related/Prior Work on MFI

- Apriori
- MaxMiner
- DepthProject
- MaxClique
- MaxEclat
- Pincer-Search
- VIPER

Outline

- Introduction
- Related Work
- Algorithmic Components
- Database Representation
- Experimental Results
- Comparison – DepthProject
- Conclusion

Algorithmic Components

- MAFIA: depth-first traversal of item subset lattice with search space pruning:
  - PEP
  - FHUT
  - HUTMFI
  - Dynamic reordering

Search Space Pruning - PEP

- Parent Equivalence Pruning
- Given current node in itemset tree with head **x** and tail element **y**, $t(x) \subseteq t(y)$ means any transaction containing **x** also contains **y**
- Since we only want **maximal** frequent itemsets, we can move **y** to the head if $t(x) \subseteq t(y)$ holds

Search Space Pruning - FHUT

- Frequent Head Union Tail
- For a node *n*, the largest possible frequent itemset contained in subtree rooted at *n* is *n's* HUT (Head Union Tail).
- If *n*'s HUT is found to be frequent, do not explore any subsets of the HUT.
- The subtree rooted at *n* can be pruned away.

Search Space Pruning - HUTMFI

- Head Union Tail Maximal Frequent Itemset
- If a superset of HUT for the current node is already in the MFI, then the HUT is frequent.
- The subtree rooted at this node can be pruned away.

Search Space Pruning – Dynamic Reordering

- Tail of a node such that it only contains frequent extensions of the current node
- Tail elements are ordered by increasing support (keeps search space as small as possible).

MAFIA Algorithm

```
PEP (Current node C, MFI) {
  Expand all children using C.tail and reorder children by
    increasing support
  Move children from C.tail to C.head with PEP pruning
  HUT = C.head union C.tail;
  if HUT is in the MFI
    Stop searching and return
  For each item i in C.tail {
    newNode = C union {i}
    IsHUT = whether i is the first item in the tail
    if newNode is frequent
      FHUT (newNode, MFI, IsHUT)
  }
  if (IsHUT and tail is frequent)
      Stop search and go back up tree
  if (C is a leaf and C.head is not in MFI)
    Add C.head to MFI
}
```

PEP

HUTMFI

FHUT

(from: http://himalaya-tools.sourceforge.net/mafiappt_files/800x600/Slide26.html)

Outline

- Introduction
- Related Work
- Algorithmic Components
- Database Representation
- Experimental Results
- Comparison – DepthProject
- Conclusion

Database Representation

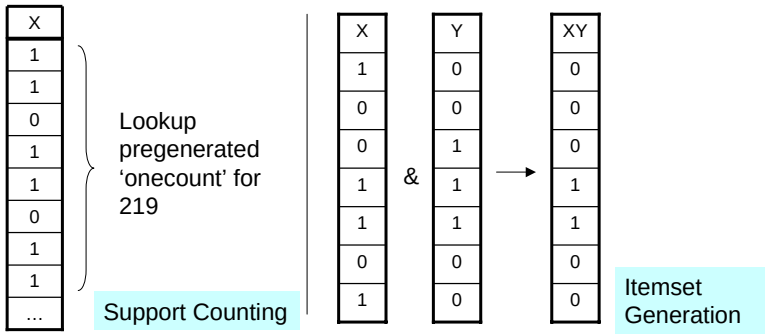
- Vertical Bitmap
- Each item is allocated a set of bits, one bit for each transaction in the database
- If item X appears in transaction j, then the j^{th} bit of item X is set to one

TID	Items
1	1, 2, 4
2	1, 2, 5
3	1, 2, 3, 4
4	1, 4

1	2	3	4	5
1	1	0	1	0
1	1	0	0	1
1	1	1	1	0
1	0	0	1	0

Database Representation

- Vertical bitmap representation allows for optimized support counting and efficient itemset generation



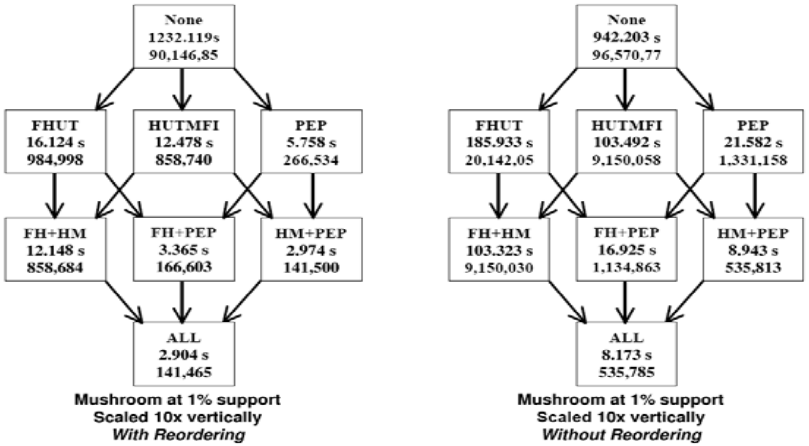
Database Compression

- Problem: Sparse bitmaps at low support levels
- Solution: Remove bits that don't matter
- To count support of subtree rooted at a node N, only need transactions containing itemset X at node N
- Product: projected bit vector

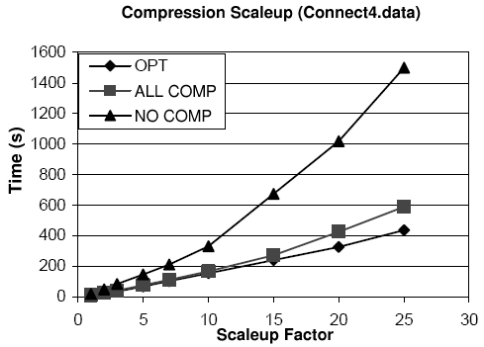
Outline

- Introduction
- Related Work
- Algorithmic Components
- Database Representation
- Experimental Results
- Comparison - DepthProject
- Conclusion

Experimental Results



Experimental Results - Compression



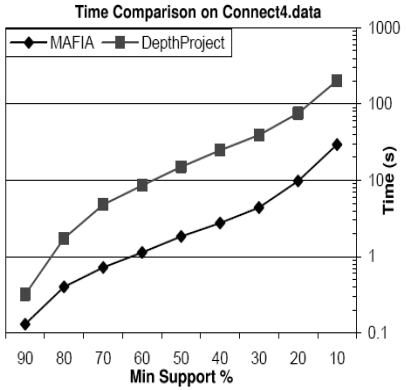
Outline

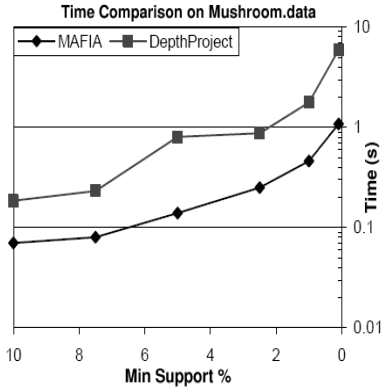
- Introduction
- Related Work
- Algorithmic Components
- Database Representation
- Experimental Results
- Comparison - DepthProject
- Conclusion

Comparison - DepthProject

- DepthProject: “state-of-the-art” maximal pattern algorithm
- Differences:
  - Uses horizontal database layout
  - Alternate pruning: bucketing

Comparison - DepthProject





Comparison - DepthProject

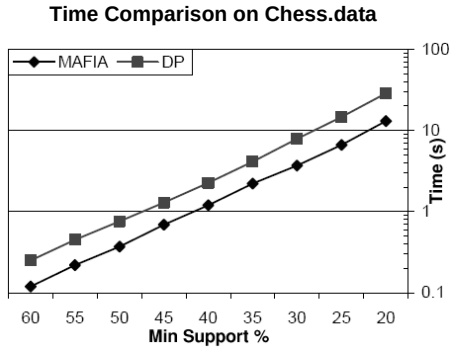
- Influence of PEP

DEPTH	1	2	3	4	5	6	7	8
Connect-4	1.23	4.22	16.46	61.07	134.8	169.06	79.98	50.43
Mushroom	1	12.64	51.81	73.15	72.92	56.81	41.29	26.78
Chess	1.33	4.14	5.73	6.51	6.73	6.7	5.97	4.95

Reduction Factor of Nodes Considered Due to PEP Pruning

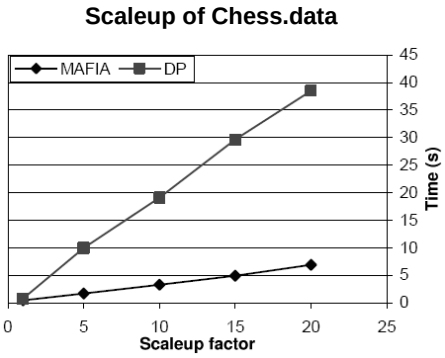
Comparison - DepthProject

- MAFIA: Only a factor of two better than DepthProject on this dataset



Comparison – DepthProject

- MAFIA: Scales very well with the number of transactions



Outline

- Introduction
- Related Work
- Algorithmic Components
- Database Representation
- Experimental Results
- Comparison – DepthProject
- Conclusion

Conclusions

- Increased efficiency of MAFIA over DepthProject due to:
  - Fast itemset generation and support counting
  - Parent-equivalence pruning

Conclusions – MAFIA flexibility

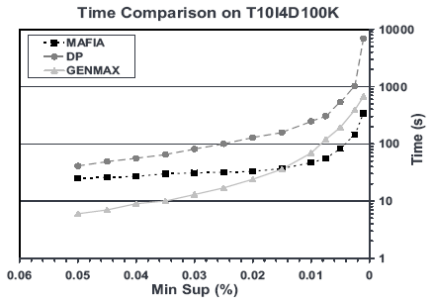
- MAFIA can also be used to find all FI
- To Find FI:
  - Suppress all pruning tools (PEP, FHUT, HUTMFI).
  - Add all frequent nodes in itemset lattice to FI without superset checking

Conclusions – MAFIA flexibility

- MAFIA can be used to mine FCI
- To find FCI:
  - Only use PEP for pruning
  - Still check for supersets in previously discovered FCI

Conclusion - Followup

- After original paper, new version of MAFIA uses *progressive focusing* technique introduced in GenMax [Gouda,Zaki]: *LMFI* update



Conclusions

- MAFIA shines when:
  - Data is dense and contains long itemsets
  - Database is large
- MAFIA is not so good when:
  - minimum support is high (short itemsets)
- MAFIA and GenMax are both useful

Thank you!

Questions?