Dynamic Itemset Counting and Implication Rules For Market Basket Data

Sergey Brin , Rajeev Motwani, Jeffrey D. Ullman, Shalom Tsur

SIGMOD'97, pp. 255-264,
Tuscon, Arizona, May 1997

Contents

Introduction

- Market Basket analysis and Association rules
- Apriori Algorithm
- The DIC algorithm
- Implication Rules vs. Association Rules

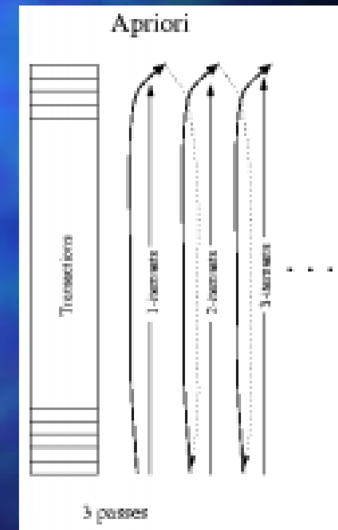
Contents

Apriori algorithm

Let L_k be the set of large k – itemsets
Let C_k be the set of candidate k-itemsets
Result := 0;
K:=1;
C_1=set of all 1-itemsets;
While C_k<> 0 do
  create a counter foe each itemset in $C_{k;}$
  forall transactions in database do
    Increment the counters of itemsets in C_k
    which occur in the transaction;
  L_k:= All candidates in C_k
  Result := Result U L_k;
  C_{k+1}:= all k+1-itemsets which have all their
    k-item subsets in $L_{k.}$
  k:=k + 1;
end

Apriori Algorithm contd.

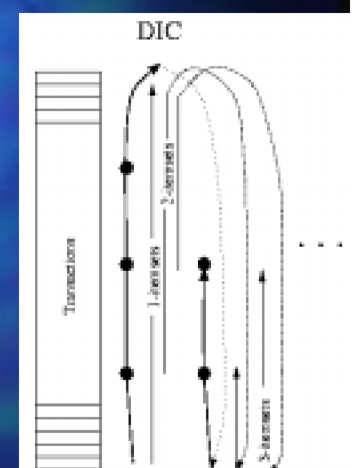
- Needs k passes to find the k-itemset
- Assumes closure property



Contents

The DIC Algorithm

- Makes use of the Closure Property
- Does not require as many passes as Apriori
- Counting can start as soon as the itemset has support



The DIC Algorithm

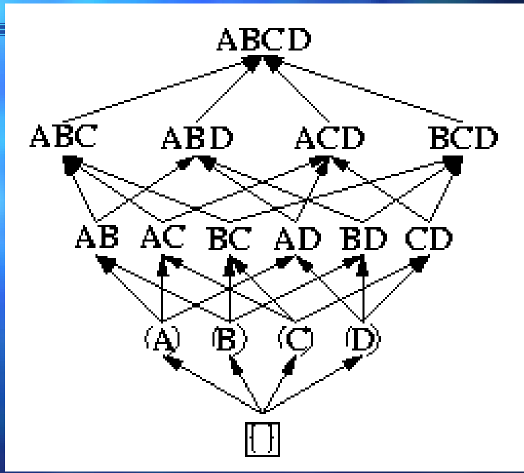
Some Notations

- Solid Box
- Solid Circle
- Dashed box
- Dashed circle

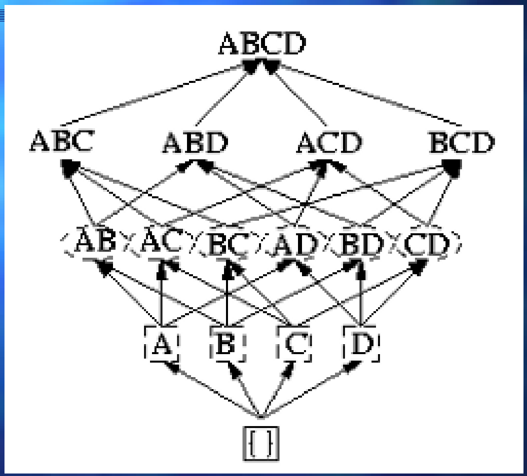
The DIC Algorithm

1. Mark empty set with a solid box. All the 1 – itemset are marked with dashed circles & others unmarked
2. Read M transactions . For each transaction increment the counter marked with dashes .
3. If a dashed circle count exceeds threshold , turn it into a dashed square . If any of the superset has all its subsets as solid or dashed square add counter and make dashed circle to superset.
4. If a dashed itemset has been counted thro' all transactions make it solid & stop counting.
5. If end of file then rewind to beginning.
6. If any more dashed items then goto step 2

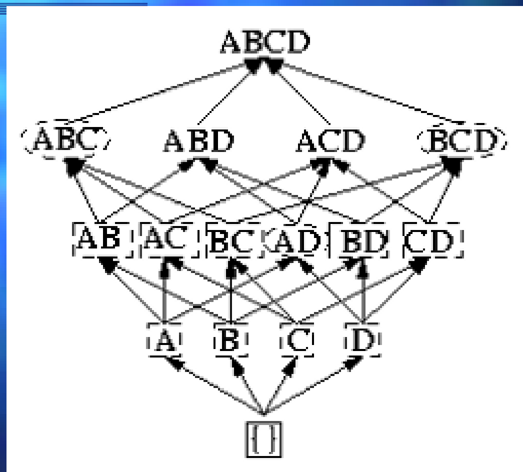
DIC Algorithm



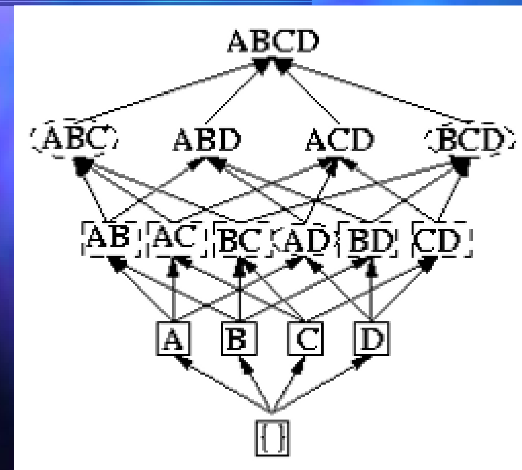
DIC Algorithm

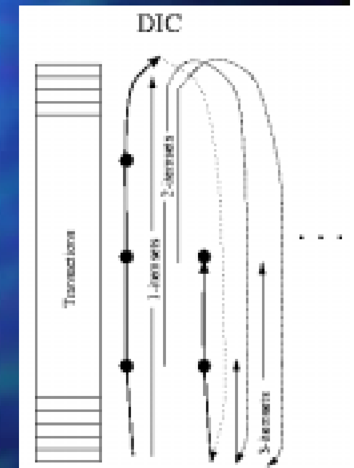


DIC Algorithm

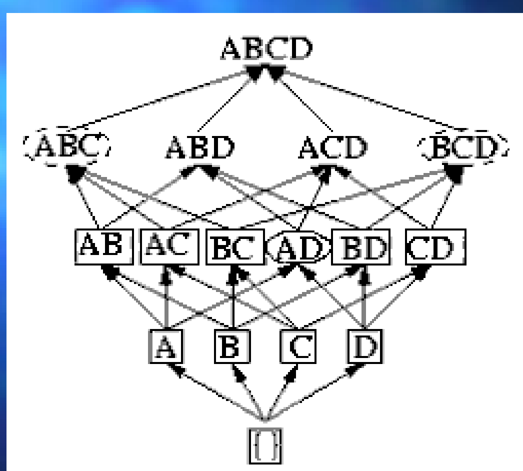


DIC Algorithm





DIC Algorithm

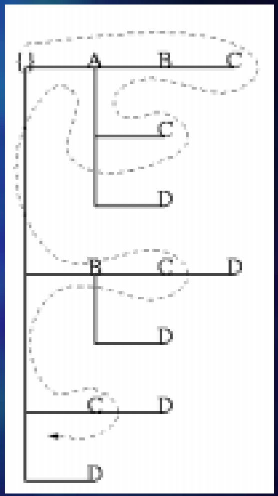


Contents

Data Structure Used

Data Structure should facilitate the following operations
1. Add new elements
2. Maintain a counter for every itemset
3. Maintain itemset states & perform transactions from dashed to solid & from circle to square
4. To determine new itemsets to be added

HASH TRIE structure is used



Contents

Concept of interest

- Confidence = P(B,A)/P(A) for A => B
  What if Confidence = P(B) ???
- Interest = P(A,B)/P(A)P(B)
- Conviction = P(A) P(~B)/P(A,~B)

How is this useful ?

1. Helps determine independence of items
2. Reduces number of rules

Contents

Advantages of DIC

1. The number of passes is less if data is homogenous
2. Has the flexibility of adding & deleting datasets on the fly
3. This algorithm can be extended to parallel versions

Disadvantages of DIC

1. Sensitivity to homogeneous data
2. Dependence on the data location

Contents

Suggestions to tackle the problems

- Virtual randomization of data
- Slacken the support threshold
- Reporting correlation of data with its location
- Item Reordering

Item Reordering

- The arrangement of the items in a transaction affects the performance
- To get the optimum cost minimize the running cost of the Increment algorithm

The Counter Increment algorithm

```
Increment(T,S) {
    /* increment this node counter*/
    T.counter++
    If T is not a leaf then for all i, 0≤i≤n
    /* increment branches as necessary*/
    If T.branches[S[i]] exists:
    Then Increment(T.branches[S[i]],(S[i+1..n])
    Return.}
```

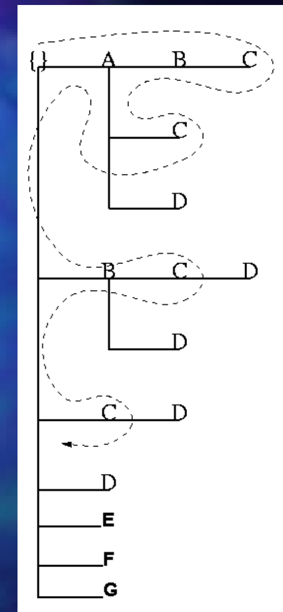

Cost Analysis

- Cost to insert ABCDEFG

∅	7
A	6
AB	5
B	5
BC	4
C	4
total	31

- Cost to insert EFGABCD

∅	7
A	3
AB	2
B	2
BC	1
C	1
total	16



Contents

Results

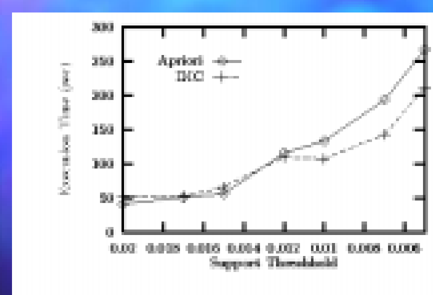


Figure 8: Performance of Apriori and DIC on Synthetic Data

Performance of Apriori and DIC On synthetic Data

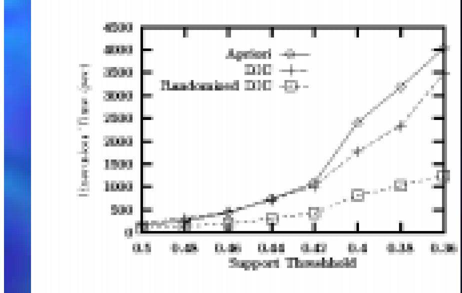
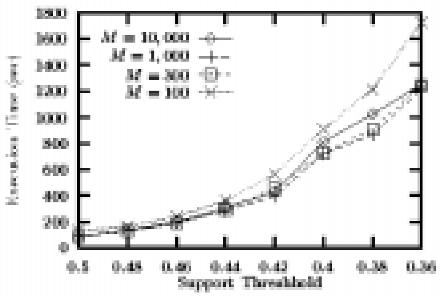
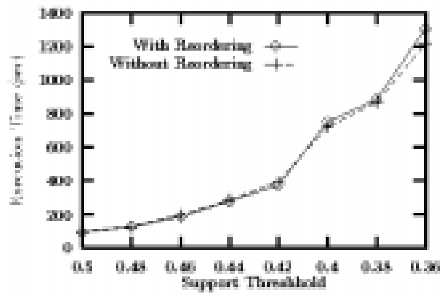


Figure 9: Performance of Apriori and DIC on Census Data

Performance of Apriori and DIC On Census Data

Results





Effect of Varying Interval Size On Performance

Performance With and Without Item Reordering

Contents

Conclusions

1. DIC especially when combined with randomization provided better performance than Apriori.
2. But reordering did not work as well as it was expected to
3. Due to the flexible and dynamic nature , it can be adapted for parallel mining & incremental mining.
4. Some Conviction values had no meaning.
5. Implication rules are made based on both the precedent and the consequence.

Contents

Topics of discussion

1. How to parallelize this algorithm ?

2. Similarity to pipelining?

3. Why is this concept not being used in many applications?