CLOSET:An Efficient Algorithm for Mining Frequent Closed Itemsets

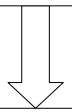
Jian Pei, Jiawei Han and Runying Mao

The shortcomings of the frequent pattern mining

- There may exist a large number of frequent itemsets in a transaction database, especially when the support threshold is low;
- There may exist a huge number of association rules. It it hard for users to comprehend and manipulate a huge number of rules.

An interesting alternative

mining the complete set of frequent itemsets and their associations.



only mining the frequent closed itemsets and their corresponding association rules.

A simple example

Transaction ID	Items in transaction
10	a1,a2,a3….a100
20	a1,a2,a3….a50

The minimum support threshold is 1;
The minimum confidence threshold is 50%

The comparison of the two mining methods

Traditional Method	FCI Method
≈10^{30} Frequent itemsets: (a1),…(a100), (a1,a2)…(a99,a100)… (a1,a2,…a100) a tremendous member of association rules…	Only two FCI: (a1, a2, …a50) (a1,a2,…a100) One association rule: (a1,a2,…a50)→ (a51,a52,…a100)

DEFINITION 1 (Frequent Closed Itemset)

- An itemset X is a closed itemset
  if there exists no itemset X' such that
  1> X' is a proper superset of X ;
  2>every transaction containing X also contains X';
- A closed itemset X is frequent
  if its support passes the given support threshold.

DEFINITION 2 (Conditional Database)

- Given a transaction database TDB. Let k be a frequent item in TDB. The k-conditional database, denoted as TDB|k, is the subset of transactions in TDB containing k, and all the occurrences of infrequent items, item k, and items following k in the f_list are omitted.

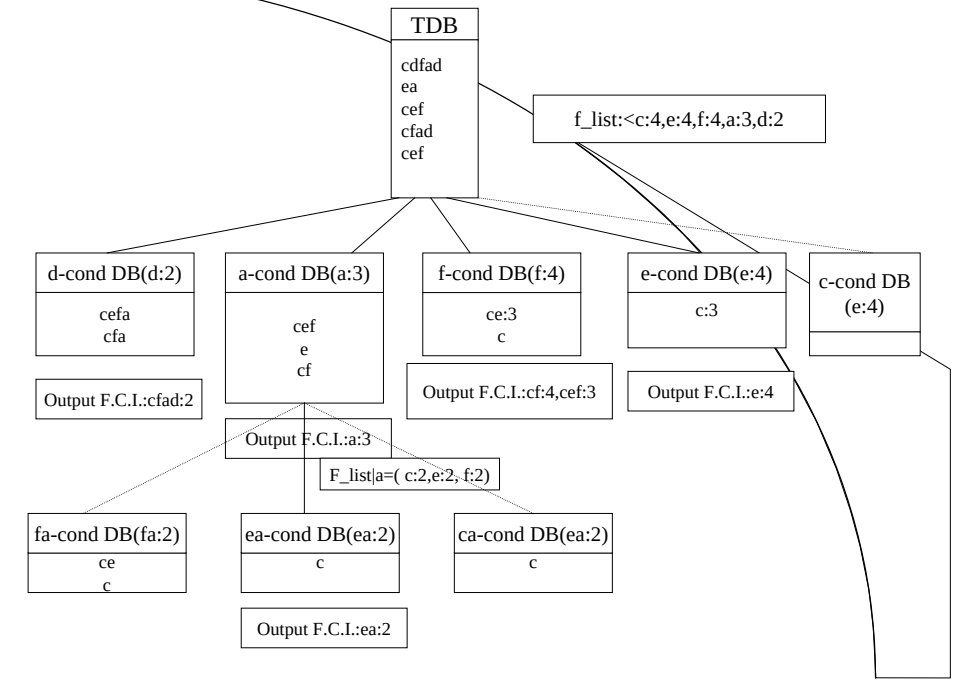
An important Lemma

- Given a transaction database TDB, a support threshold min_sup, and f_list=(i1,i2,…,in), the problem of mining the complete set of frequent closed itemsets can be divided into n sub-problems: The j_{th} problem($1 \le j \le n$) is to find the complete set of frequent closed itmesets containing i_{n+1-j} but no i_k (for $n+1-j < k \le n$)

The transaction database TDB

Transaction ID	Items in transaction
10	a,c,d,e,f
20	a,b,e
30	c,e,f
40	a,c,d,f
50	c,e,f

Min_sup=2



Optimization 1

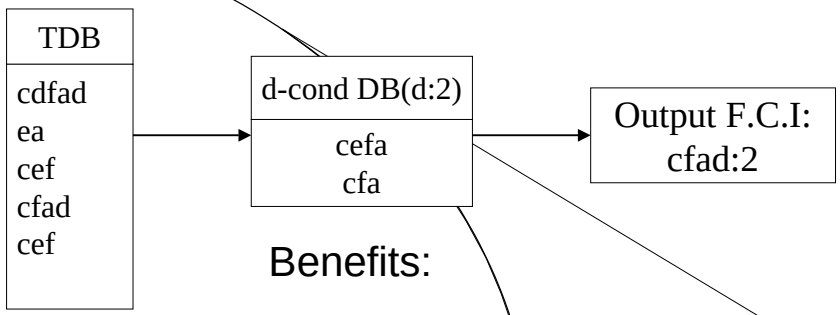
Compress transactional and conditional database using an FP-tree structure

Benefits

- FP-tree compresses database for frequent itemset mining.
- Conditional databases can be derived from FP-tree efficiently.

Optimization 2

Extract items appearing in every transaction of conditional database



Benefits:

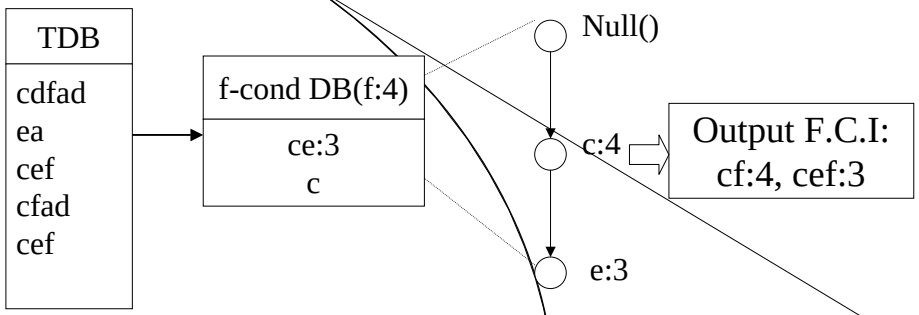
- It reduces the size of FP-tree;
- It reduces the level of recursions.

Lemma 2

- If an itemset Y is the maximal set of items appearing in every transaction in the X-conditional database, and X ∪ Y is not subsumed by some already found frequent closed itemset with identical support, then X ∪ Y is a frequent closed itemset.

Optimization 3

Directly extract frequent closed itemsets from FP-tree



DEFINITION 3 (k-single segment itemsets)

- Let k be a frequent item in the X-conditional database. If there is only one node N labeled k in the corresponding FP-tree, every ancestor of N has only one child and N has (1)no child, (2)more than one child, or (3)one child with count value smaller than that of N, then the k-single segment itemset is the union of itemset X and the set of items including N and N's ancestors(excluding the root).

Lemma 3

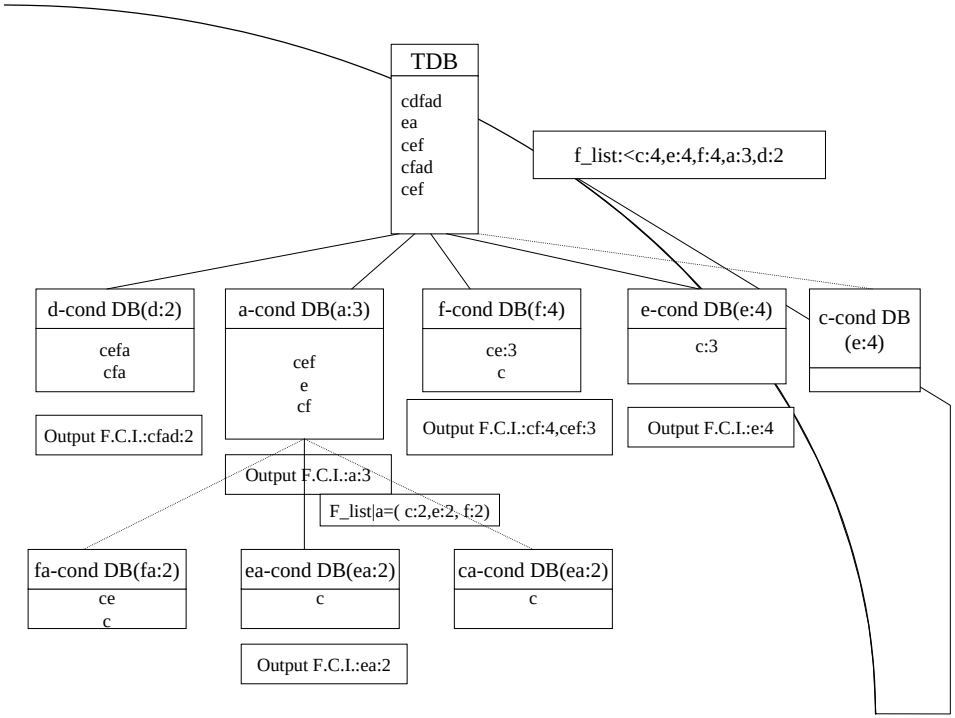
- The i_single segment itemset Y is a frequent closed itemset if the support of i within the conditional database passes the given threshold and Y is not a proper subset of any frequent closed itemset already found.

Optimization 4

Prune search branches

Lemma 4

Let X and Y be two frequent itemsets with the same support. If $X \subset Y$, and Y is closed, then there exist no frequent closed itemset containing X but not Y-X



The Algorithm of CLOSET

- Initialization. Let FCI be the set of frequent closed itemset. Initialize 0→FCI;
- Find frequent items. Scan transaction database TDB, compute frequent item list;
- Mine frequent closed itemsets recursively. Call CLOSET(0, TDB, f_list, FCI).

Subroutine CLOSET(X,DB,f_list,FCI)

- 1.Let Y be the set of items in f_list such that they appear in every transaction of DB, insert $X \cup Y$ to FCI if it is not a proper subset of some itemset in FCI with same support;//Applying Optimization2
- 2.Build FP-tree for DB, items already be extracted should be excluded;//Applying Optimization1
- 3.Apply Optimization3 to extract frequent closed itemsets if it is possible;
- 4.Form conditional database for every remaining item in f_list, at the same time, compute local frequent item lists for these conditional databases;

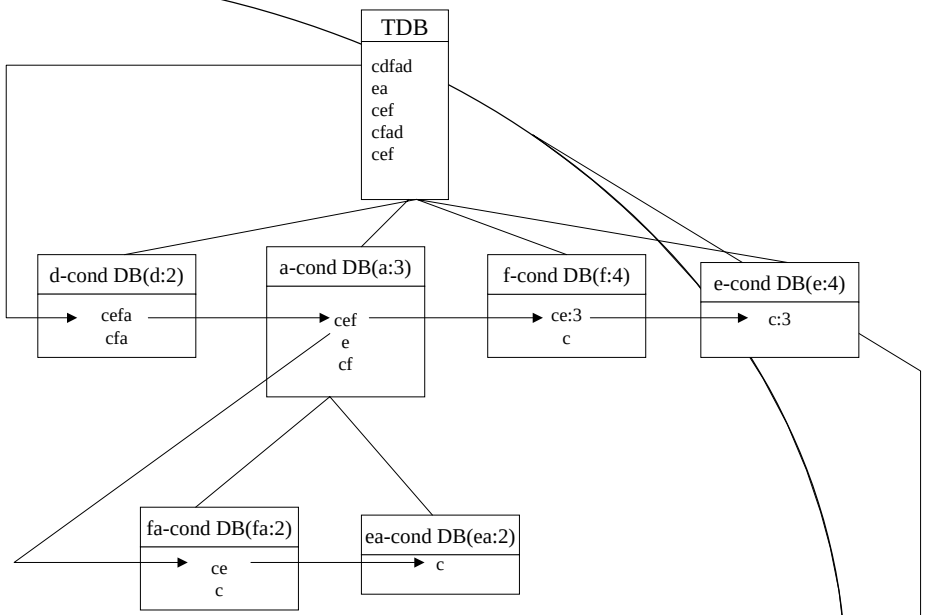
Subroutine CLOSET(X,DB,f_list,FCI)

- 5.For each remaining item I in f_list, starting from the last one, call CLOSET(iX, $DB|_i$, f_list_i, FCI). If iX is not a subset of any frequent closed itemset already found with the same support count, where $DB|_i$ is the i-conditional database with respect to DB and f_list is the corresponding frequent item list.//Applying Optimization4

Scaling up CLOSET in large database

When the transaction database is large, it is unrealistic to construct a main memory-based FP-tree.

- Construct conditional database without FP-tree
- Construct disk-based FP-tree



Performance Study

Reduction of the szie of itemsets

Support	#F.C.I	#F.I	$\frac{\#F.I}{\#F.C.I}$
64179(95%)	812	2,205	2.72
60801(90%)	3,486	27,127	7.78
54046(80%)	15,107	533,975	35.35
47290(70%)	35,875	4,129,839	115.12

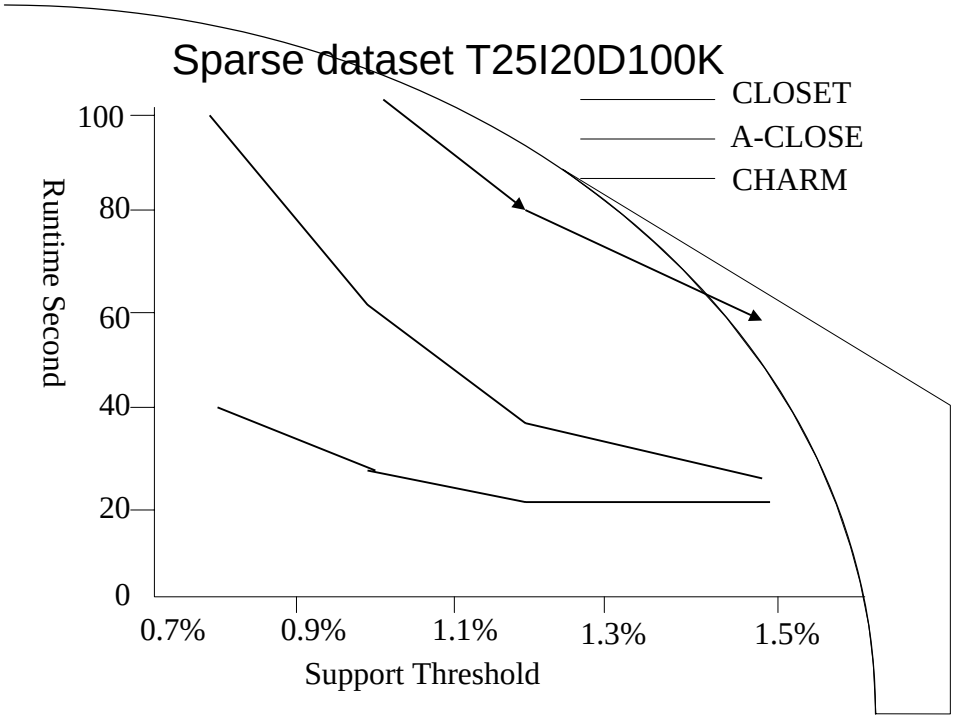
Performance Study

A-close and CHAEM

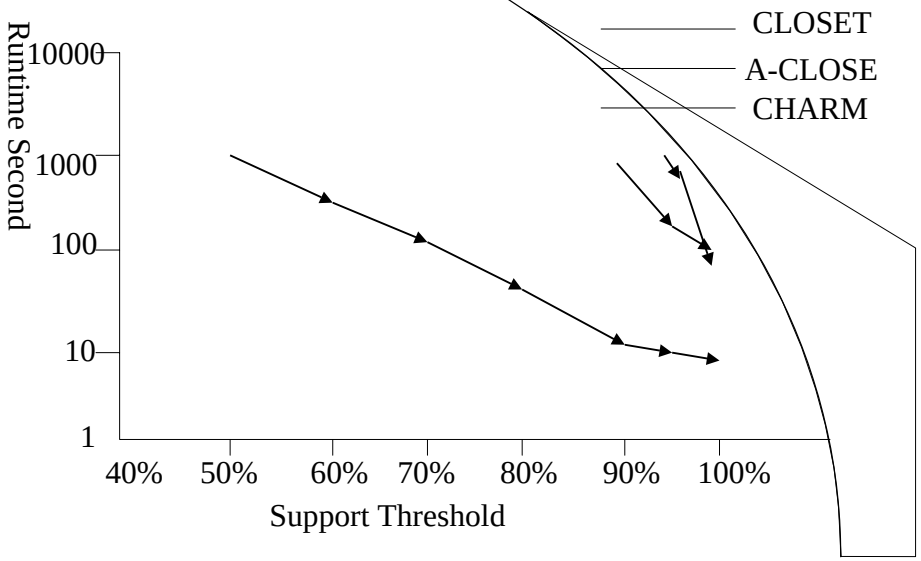
vs

CLOSET

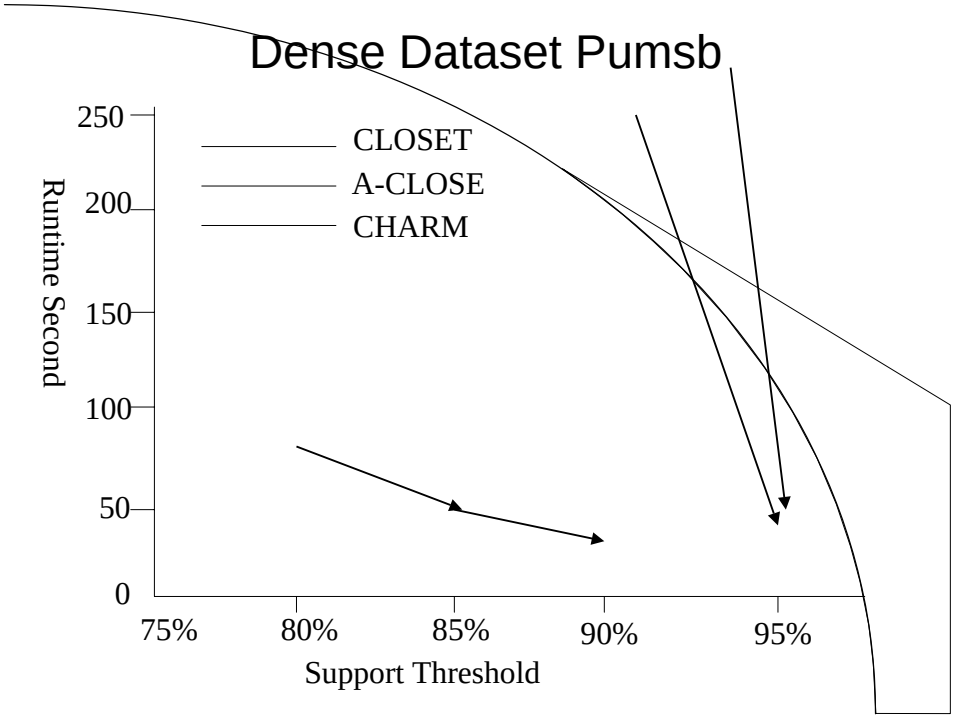
Sparse dataset T25I20D100K

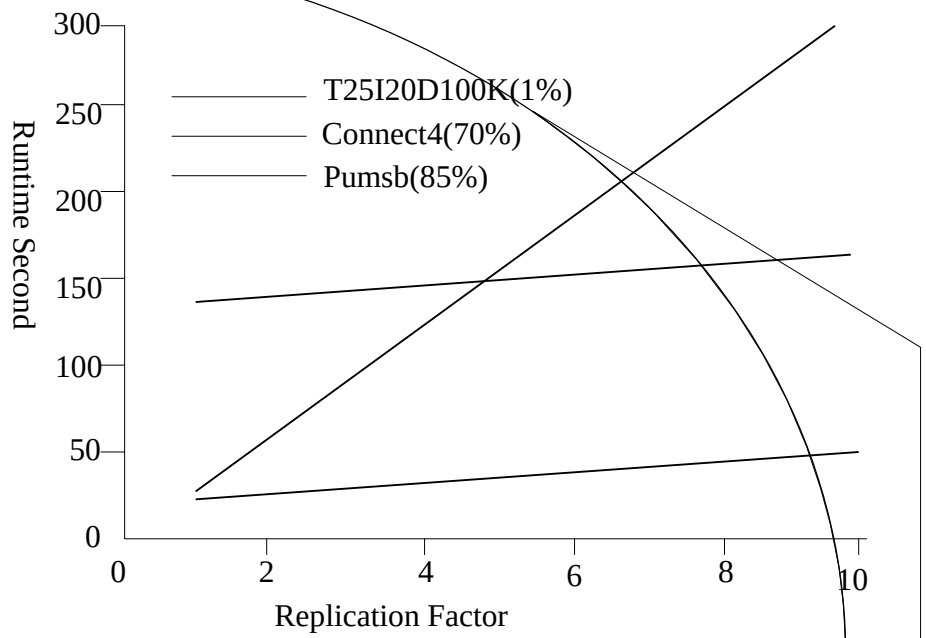


Dense Dataset Connect-4



Dense Dataset Pumsb





Conclusions

Three techniques:

- Applying a compressed FP-tree structure;
- Developing a single prefix path compression technique;
- Exploring a partition-based projection mechanism.

Thank You!