Efficient Mining of Multiple Fuzzy Frequent Itemsets

French-Azerbaijani University

Ali Khudiyev Nurlan Imanov Hafiz Gahramanov

Introduction

- Traditional association-rule mining or frequent itemset mining only can handle **binary database**(**0** or **1**).

- In real-life situations, quantitative databases can be used to provide more information for **decision making** than that of traditional **binary databases**.

- However, it is difficult to handle quantitative databases based on **crisp sets**.

- Fuzzy set theory , which was proposed to handle quantitative databases, is based on **pre-defined membership function** to transform the quantitative values into the representation of **linguistic terms**.

The reviewed paper proposes

- **MFFI-Miner algorithm:**
  - No candidate generation
  - Based on fuzzy-list structure
  - Reduces complexity of generate-and-test approach in a level-wise manner
  - With 2 pruning strategies to reduce the search space of tree based on the above-mentioned structure
  - Computation can be greatly reduced
  - Better performance than Apriori-based and pattern-growth algorithms

Preliminaries

- D = { T1, T2, ..., Tn } - quantitative database
- I = { i1, i2, ..., im } - finite set of m distinct items

- Tq is included in D :
  - subset of I
  - format: items and purchase quantities v[iq]
  - represented by a unique identifier TID

- X is an itemset of k distinct items { i1, i2, ..., ik } and called k-itemset.
- minsup - minimum support threshold (initially set as 25%)
- Memb - a set of user-specified membership functions

Note: ‘**Memb**’ and ‘**minsup**’ are user-specified.

An example of a quantitative database and transformed results

TID	Items and their quantities	Transformed linguistic terms
1	(C:3), (D:2), (E:1)	$\frac{0.67}{C.M}+\frac{0.5}{C.H}, \frac{0.5}{D.L}+\frac{0.67}{D.M}, \frac{1.0}{E.L}$
2	(B:1), (C:2), (D:1)	$\frac{1.0}{B.L}, \frac{0.5}{C.L}+\frac{0.67}{C.M}, \frac{1.0}{D.L}$
3	(B:3), (C:3), (E:1)	$\frac{0.67}{B.M}+\frac{0.5}{B.H}, \frac{0.67}{C.M}+\frac{0.5}{C.H}, \frac{1.0}{E.L}$
4	(A:3), (C:5), (D:3)	$\frac{0.67}{A.M}+\frac{0.5}{A.H}, \frac{1.0}{C.H}, \frac{0.67}{D.M}+\frac{0.5}{D.H}$
5	(A:1), (B:1), (C:2), (D:1)	$\frac{1.0}{A.L}, \frac{1.0}{B.L}, \frac{0.5}{C.L}+\frac{0.67}{C.M}, \frac{1.0}{D.L}$
6	(B:1), (D:1), (E:2)	$\frac{1.0}{B.L}, \frac{1.0}{D.L}, \frac{0.5}{E.L}+\frac{0.67}{E.M}$
7	(A:4), (B:3), (D:5), (E:3)	$\frac{1.0}{A.H}, \frac{0.67}{B.M}+\frac{0.5}{B.H}, \frac{1.0}{D.H}, \frac{0.67}{E.M}+\frac{0.5}{E.H}$
8	(B:1), (C:2), (D:1)	$\frac{1.0}{B.L}, \frac{0.5}{C.L}+\frac{0.67}{C.M}, \frac{1.0}{D.L}$

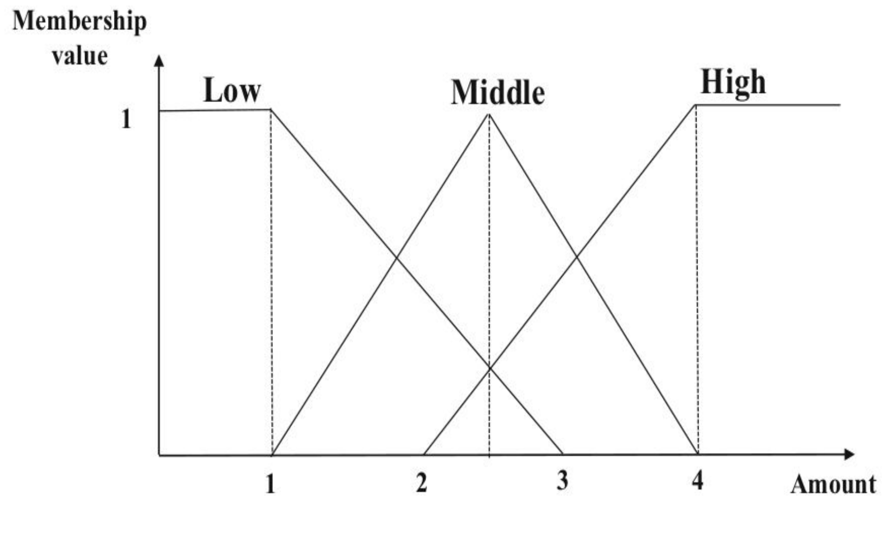


Fig. 1 The used membership functions of fuzzy linguistic 3-terms

$$f_{iq} = \mu_i(v_{iq}) \left(= \frac{fv_{iq1}}{R_{i1}} + \frac{fv_{iq2}}{R_{i2}} + \cdots + \frac{fv_{iqh}}{R_{ih}} \right),$$

V_{iq} → quantitative value of i in q-th transaction(ex: C → 3 when q == 1)

Problem Statement

- **Goals :**
  - Speed up the mining process
  - Discover the complete set of MFFIs
  - MFFI <- { X | supp(X) is greater than or equal to minsup*|D| } D → is the database size

$$\{X \in R_{il} | \sum_{X \subseteq T_q \wedge T_q \in D'} \min(fv_{aql}, fv_{bql}), a, b \in X, a \notin b\}$$

Proposed MFFI-Miner Algorithm

Phases :

1. Transformation

2. Fuzzy-list construction

3. Search space of enumeration tree

Transformation Phase

- Transformation of quantitative value of each linguistic variable(item) into several fuzzy linguistic terms(fuzzy itemsets)
- Introducing membership degrees(fuzzy values) of the fuzzy itemsets
- Support of a fuzzy itemset is the summation of all fuzzy values of the same fuzzy itemset
- If support is no less than the minsup count then the fuzzy itemset is FFI (kept transformed)
- Sorting remaining fuzzy itemsets with their fuzzy values in support-ascending order
    - To perform intersection operation

Fuzzy-list construction Phase

- Definitions:
  - Tq/R[il] is to indicate the set of fuzzy itemsets after R[il] in Tq
  - The fuzzy value of R[il] is evaluated as if(R[il], Tq) ("if" stands for internal fuzzy [value])
  - The resting fuzzy value except R[il] in Tq is evaluated as rf(R[il], Tq) = max(if(z,Tq) | z in (T1/R[il]) ("rf" stands for resting fuzzy [value])

Algorithm 1: Fuzzy-list Construction

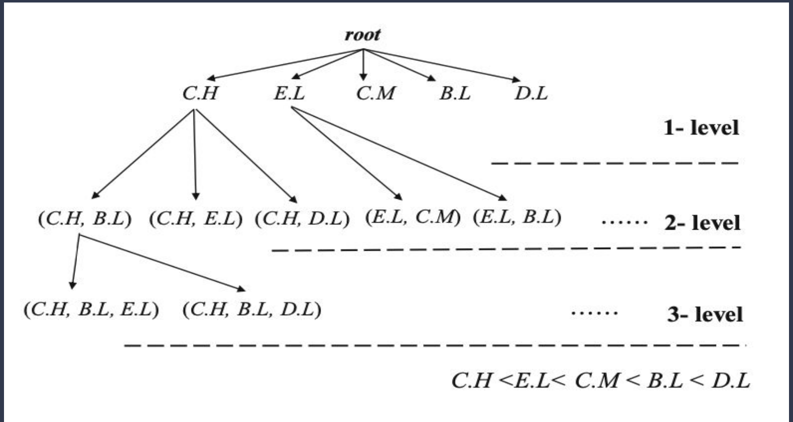
Input: $P_x.FL$, the fuzzy-list of P_x; $P_y.FL$, the fuzzy-list of P_y.
Output: $P_{xy}.FL$, the fuzzy-list of x and y.

if *P_x and P_y belong to the same item* **then**
  return *null*.
$P_{xy}.FL \leftarrow null$;
for *each $E_x \in P_x.FL$* **do**
  if *$\exists E_y \in P_y.FL$ and $E_x.tid == E_y.tid$* **then**
    $E_{xy}.tid \leftarrow E_x.tid$;
    $E_{xy}.if \leftarrow min(E_x.if, E_y.if)$;
    $E_{xy}.rf \leftarrow E_y.rf$;
    $E_{xy} \leftarrow < E_{xy}.tid, E_{xy}.if, E_{xy}.rf >$;
    append E_{xy} to $P_{xy}.FL$.
return $P_{xy}.FL$.

- Note :
  - SUM.R[il].if = sum of all if(R[il], Tq] over the transformed database D'
  - SUM.R[il].rf = sum of all rf(R[il], Tq] over the transformed database D'

Search space of enumeration tree

- Input: fuzzy-list of 1-itemset and minsup %
- Output: MFFIs
- Pruning of itemsets; if an itemset is not frequent then all of its supersets is not frequent either
- DFS approach (with recursive calls)



Algorithm 2: MFFI-Miner

Input: FLs, fuzzy-list of 1-itemsets; δ.
Output: $MFFIs$, the set of multiple fuzzy frequent itemsets.

```
for each fuzzy-list X in FLs do
    if SUM.X.if ≥ δ × |D| then
        MFFIs ← X ∪ MFFIs.
    if SUM.X.rf ≥ δ × |D| and SUM.X.if ≥ δ × |D| then
        exFLs ← null;
        for each fuzzy-list Y after X in FLs do
            exFL ← exFLs + Construct(X, Y);
        MFFI-Miner(exFLs, δ);
return MFFIs.
```

C.H		
1	0.5	0.67
3	0.5	0.67
4	1	0

E.L		
1	1	0.67
3	1	0.67
6	0.5	1

C.M		
1	0.67	0.5
2	0.67	1
3	0.67	0
5	0.67	1
7	0.67	1

B.L		
2	1	1
5	1	1
6	1	1
7	1	1

D.L		
1	0.5	0
2	1	0
5	1	0
6	1	0
7	1	0

tid if rf

Experimental Results

Metrics:

- Runtime
- Memory Usage
- Node analysis

Used algorithms:

- GDF
- UBMFFP
- MFFI-Miner[PI,PR]

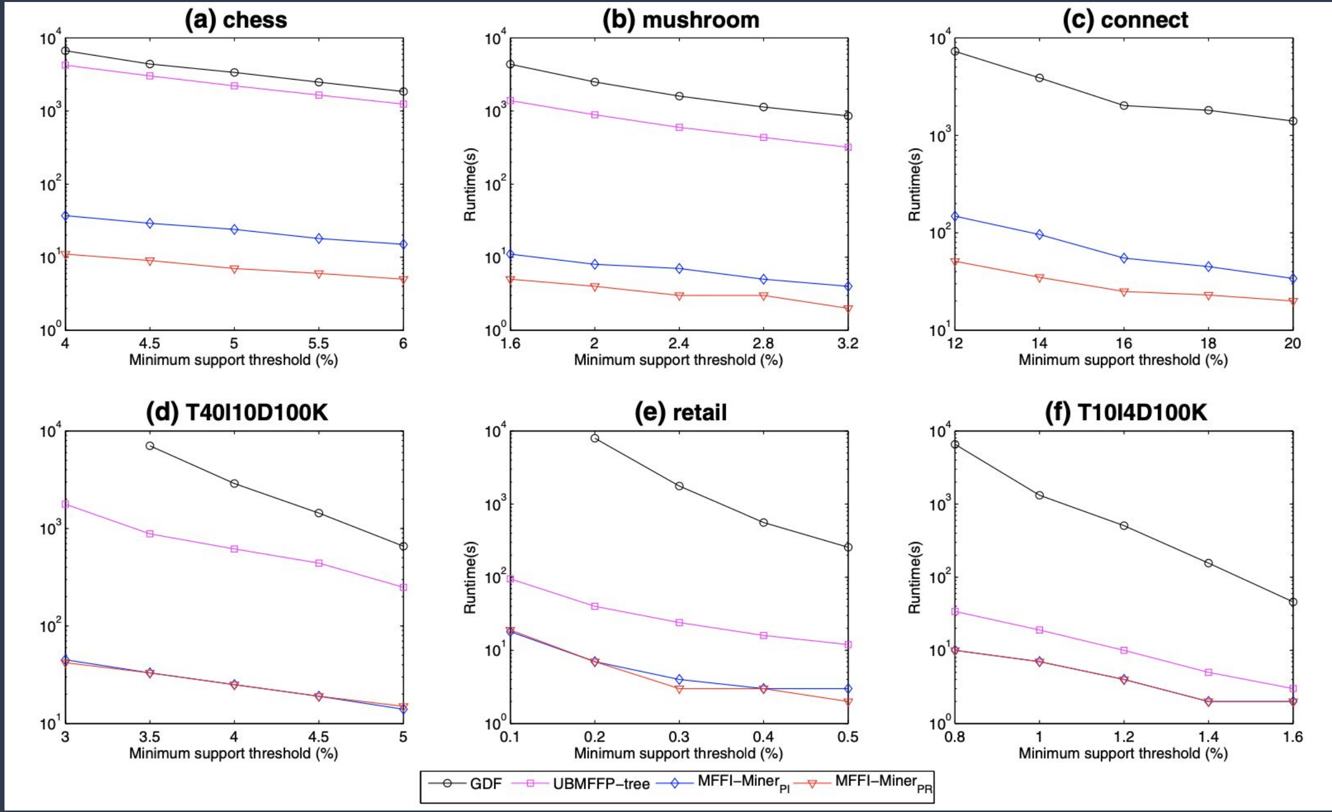
Datasets:

- Four real-life chess
- Mushroom
- Connect
- Retail
- T10I10D100k & T10I4D100k (synthetic)

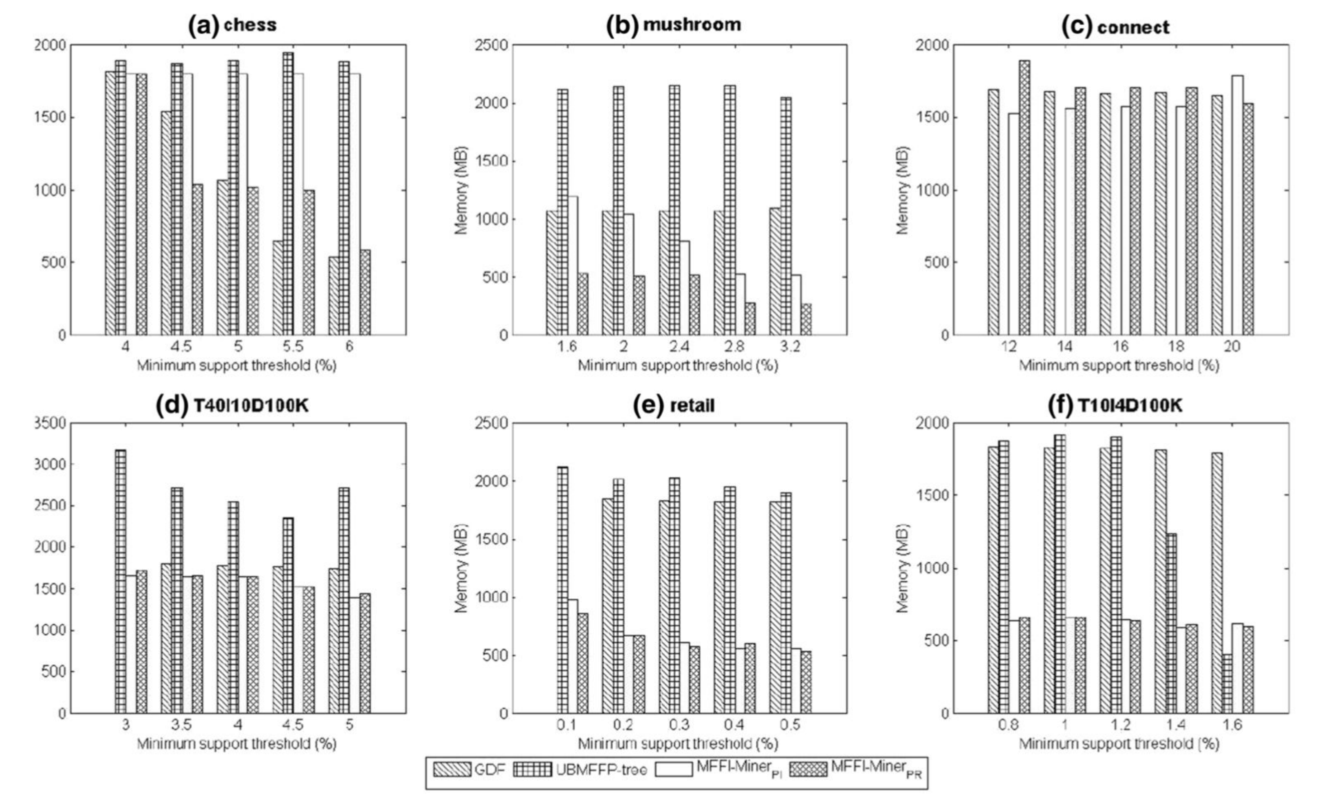
Process:

- Quantities of items were randomly assigned in the range of [1, 5]
- The quantitative datasets were transformed into several fuzzy linguistic terms (based on the predefined membership function)
- The algorithm was terminated if the execution time exceeded 1000s

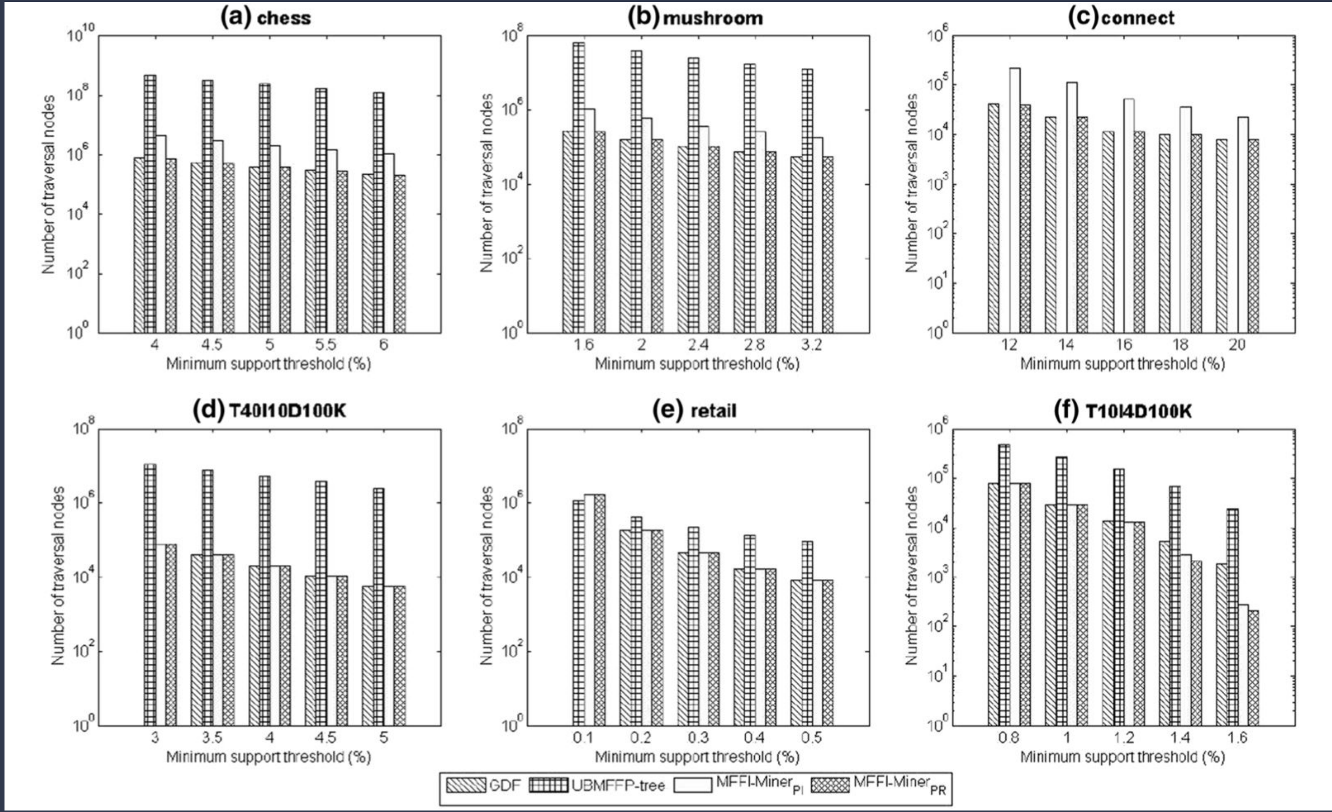
Experimental Results: Runtime



Experimental Results: Memory usage



Experimental Results: Node Analysis



Conclusion

- Two pruning strategies were designed to reduce the search space

- The proposed MFFI-Miner algorithm outperformed the GDF & UBMFFP-tree algorithms in terms of
    - Runtime
    - Memory usage
    - Number of determining candidates in both real-world and synthetic datasets

Thank You !

Ali Khudiyev Nurlan Imanov Hafiz Gahramanov