Maximal Frequent Itemset Mining

Oskar Gross
Karl Blum
Maximal Frequent Itemsets

- Has no superset that is frequent
- \( \text{MFI} \subseteq \text{FI} \)
- Example:
  - Items: \( a, b, c, d, e \)
  - Frequent Itemset: \( \{a, b, c\} \)
  - \( \{a, b, c, d\}, \{a, b, c, e\}, \{a, b, c, d, e\} \) are not Frequent Itemset.
  - Maximal Frequent Itemsets: \( \{a, b, c\} \)
Lexicographic ordering

<table>
<thead>
<tr>
<th>Transaction</th>
<th>Head</th>
<th>Tail</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>1</td>
<td>2,3,4</td>
</tr>
<tr>
<td>G</td>
<td>2,3</td>
<td>4</td>
</tr>
</tbody>
</table>
MAFIA

• 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

• Depth first traversal

• Effective pruning
Algorithmic Components

• Depth-first traversal
• Search space pruning
  – PEP
  – FHUT
  – HUTMI
  – 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).
GenMax

• Optimizing superset checking techniques

• Reordering the combine set

• Optimizing GenMax

• Final GenMax algorithm
Optimizing superset checking techniques

- Double checking problem

```plaintext
// Invocation: MFI-backtrack(∅, F₁, 0)

MFI-backtrack(Iᵢ, Cᵢ, l)
1. for each x ∈ Cᵢ
2.   Iᵢ₊₁ = Iᵢ ∪ {x}
3.   Pᵢ₊₁ = {y : y ∈ Cᵢ and y > x}
4.   if Iᵢ₊₁ ∪ Pᵢ₊₁ has a superset in MFI
5.   return //all subsequent branches pruned!
6.   Cᵢ₊₁ = FI-combine (Iᵢ₊₁, Pᵢ₊₁)
7.   if Cᵢ₊₁ is empty
8.   if Iᵢ₊₁ has no superset in MFI
9.   MFI = MFI ∪ Iᵢ₊₁
10. else MFI-backtrack(Iᵢ₊₁, Cᵢ₊₁, l + 1)
```

Maximum position $p$

Use the $p$ for 8*
Redundant subset checks elimination

• Let us have a tail with cardinality of $m$

• We should check only if MFI changes

• check_status flag

• When $C_{l+1}$ is empty check_status = true
Local Maximal Frequent Itemsets

• Only potential supersets of candidates

• Only maximal sets which contain \( I \)
Algorithm With Local Maximum Frequent Itemset

- Line 6* - initializes next iteration LMFI
- Line 11* - creates new LMFI containing x
- Line 13* - sums up the LMFI’s

```java
// Invocation: LMFI-backtrack(∅, F₁, ∅, 0)
// LMFIₗ is an output parameter

LMFI-backtrack(Iₗ, Cₗ, LMFIₗ, l)
1.   for each x ∈ Cₗ
2.     Iₗ₊₁ = Iₗ ∪ {x}
3.     Pₗ₊₁ = {y : y ∈ Cₗ and y > x}
4.     if Iₗ₊₁ ∪ Pₗ₊₁ has a superset in LMFIₗ
5.       return //subsequent branches pruned!
6.   *   LMFIₗ₊₁ = ∅
7.   Cₗ₊₁ = FI-combine (Iₗ₊₁, Pₗ₊₁)
8.   if Cₗ₊₁ is empty
9.     if Iₗ₊₁ has no superset in LMFIₗ
10.    LMFIₗ = LMFIₗ ∪ Iₗ₊₁
11.   *   else LMFIₗ₊₁ = {M ∈ LMFIₗ : x ∈ M}
12.    LMFI-backtrack(Iₗ₊₁, Cₗ₊₁, LMFIₗ₊₁, l + 1)
13.   *   LMFIₗ = LMFIₗ ∪ LMFIₗ₊₁
```
Frequency Testing Optimization

- Diffset optimization for fast frequency testing

```c
// Can \( I_{l+1} \) combine with other items in \( C_l \)?

\textbf{FI-diffset-combine}(I_{l+1}, P_{l+1})

1. \( C = \emptyset \)
2. \textbf{for each} \( y \in P_{l+1} \)
3. \( y' = y \)
4. \textbf{if} level == 0 \textbf{then} \( d(y') = t(I_{l+1}) - t(y) \)
5. \textbf{else} \( d(y') = d(y) - d(I_{l+1}) \)
6. \textbf{if} \( \sigma(y') \geq \text{min\_sup} \)
7. \( C = C \cup \{y'\} \) //add \( y' \) in increasing order of support
8. \textbf{return} \( C \)
Example 4. Suppose, that we have the itemset $ADT$; we show how to get its support using the diffset propagation technique. In GenMax we start with item $A$, and extend it with item $D$. Now in order to find the support of $AD$ we first find the diffset for $AD$, denoted $d(AD) = t(A) - t(D)$ and then calculate its support as $\sigma(AD) = \sigma(A) - |d(AD)|$. At the next level, we need to compute the diffset for $ADT$ using the diffsets for $AD$ and $AT$, where $I_1 = \{A\}$ and $C_1 = \{D,T\}$. The diffset of itemset $ADT$ is given as $d(ADT) = d(AT) - d(AD)$, and its support is given as $\sigma(AD) - |d(ADT)|$ (Zaki and Gouda, 2003). Since longer patterns are always formed by combining its lexicographic first two subsets, which share the same prefix, the method is guaranteed to be correct.

- Ehk:
  - Leiame kõik transactionid, kus D ei esine
  - Lahutame A suppordist kõik, kus D ei esine
  - Vastuseks saame suppordi, kus esinevad AD koos
GenMax Algorithm in Full Glory

\[
\text{GenMax:}
\]
1. Compute \( F_1 \) and \( F_2 \)
2. Compute \( IF(x) \) for each item \( x \in F_1 \)
3. Sort \( F_1 \) (decreasing in \( IF(x) \), increasing in \( \sigma(x) \))
4. \( \text{MFI} = \emptyset \)
5. \( \text{LMFI-backtrack}(\emptyset, F_1, \text{MFI}, 0) \) //use diffsets
6. \( \text{return MFI} \)
## Stats

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Transactions</th>
<th>Items</th>
<th>Average Transaction Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chess</td>
<td>3196</td>
<td>76</td>
<td>37</td>
</tr>
<tr>
<td>Connect4</td>
<td>67,557</td>
<td>130</td>
<td>43</td>
</tr>
<tr>
<td>pumbsb</td>
<td>49,046</td>
<td>7,117</td>
<td>74</td>
</tr>
</tbody>
</table>
Time Comparison on Chess

- MAFIA
- DP
- GEMAX

Min Sup (%) vs Time (s)