Mining Cross-Level High Utility Itemsets

Philippe Fournier-Viger[1(✉)], Ying Wang[2], Jerry Chun-Wei Lin[3], Jose Maria Luna[4], and Sebastian Ventura[4]

[1] School of Natural Sciences and Humanities, Harbin Institute of Technology (Shenzhen), Shenzhen, China
philfv@hit.edu.cn

[2] School of Computer Sciences and Technology, Harbin Institute of Technology (Shenzhen), Shenzhen, China
iwangying_919@163.com

[3] Department of Computing, Mathematics and Physics, Western Norway University of Applied Sciences (HVL), Bergen, Norway
jerrylin@ieee.org

[4] Department of Computer Sciences, University of Cordoba, Cordoba, Spain
{jmluna,sventura}@uco.es

Frequent Itemset Mining

Input:

A transaction database

TID	Items
T_1	a, c
T_2	e
T_3	a, b, c, d, e
T_4	b, c, d, e
T_5	a, c, d
T_6	a, c, e
T_7	b, c, e

a $minsup$ threshold

Output:

Frequent itemsets (with support≥$minsup$)

if $minsup = 3$, the frequent itemsets are:

{a}:4	{b,e}:3
{b}:3	{c,d}:3
{d}:3	{c,e}:4
{a,c}:4	{b,c,e}:3
{b,c}:3	

Algorithms:

- Apriori（VLDB 1994）
- Apriori-TID（VLDB 1994）
- Eclat（TKDE 2000）
- FP-Growth（ACM SIGMOD 2000）

High Utility Itemset Mining

Input:

A transaction database

TID	Items
T_1	(a,1), (c,1)
T_2	(e,1)
T_3	(a,1), (b,5), (c,1), (d,3), (e,1)
T_4	(b,4), (c,3), (d,3), (e,1)
T_5	(a,1), (c,1), (d,1)
T_6	(a,2), (c,6), (e,2)
T_7	(b,2), (c,2), (e,1)

External utility values

Item	Unit profit
a	5$
b	2$
c	1$
d	2$
e	3$

a *minutil* threshold

Output:

High-utility itemsets (with utility≥*minutil*)

if *minutil* = 30$, the *HUIs* are:

{b,d}:30$	{a,c}:34$
{b,e}:31$	{b,d,e}:36$
{a,c,e}:31$	{b,c,e}:37$
{b,c,d}:34$	{b,c,d,e}:40$

Previous work

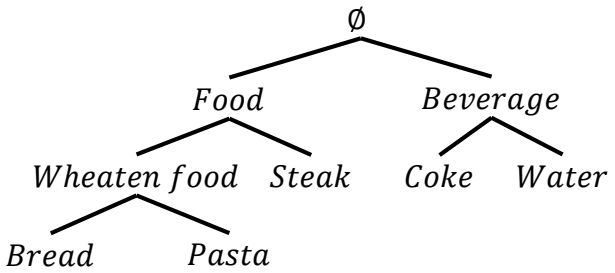
- **Several algorithms:**
  - Two-Phase (PAKDD 2005)
  - IHUP (TKDE, 2010),
  - UP-Growth (KDD 2011),
  - HUI-Miner (CIKM 2012),
  - FHM (ISMIS 2014)
  - EFIM (KAIS 2017)
  - mHUIMiner (PAKDD 2017)
- **Key idea:**

  Calculate an upper-bound on the utility of itemsets (**e.g.** the TWU) that is anti-monotonic to be able to prune the search space.

$$u(\{b,c,d\}) = u(\{b,c,d\}, T_3) + u(\{b,c,d\}, T_4)$$

$$TWU(\{b,c,d\}) = u(T_3) + u(T_4)$$

Limitation



- **High utility itemset mining**
  - is useful for discovering profitable itemsets in a whole database
  - it ignores that items are organized according to **taxonomies**.

- **Consider taxonomy information**
  - Cross-level frequent itemset mining: Cumulate, Prutax, SET
  - Multi-level frequent itemset mining: Han & Fu, Ong et al., Pramudiono
  - Cross-level frequent itemset mining using multiple thresholds: Lui & Chung
  - Multi-level frequent itemset mining using multiple thresholds: Rajkumar et al.
  - **Multi-level high utility itemset mining: MLHUI-Miner**

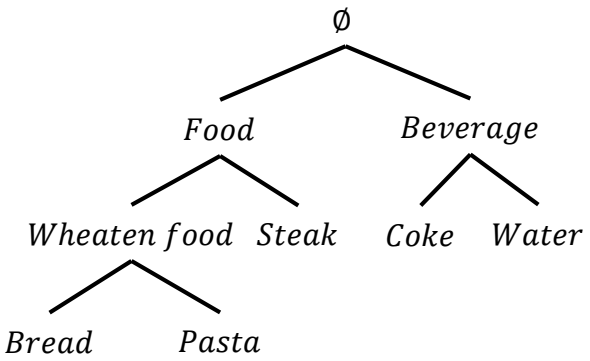
Limitation

- **MLHUI-Miner**
  - ➢ cannot find cross-level patterns
  - ➢ mines each taxonomy levels independently
  - ➢ does not use relationships between taxonomy levels

- We propose a new pattern type:

Cross-Level High Utility Itemset (CLHUI)

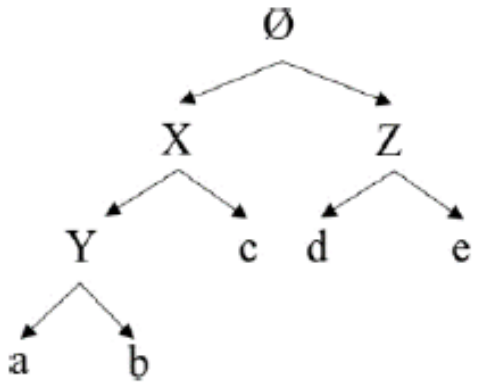
e.g. {Bread, Steak, Beverage} is a CLHUI, while not being a MLHUI or a HUI.



How to calculate the utility?

TID	Items
T_1	(a,1), (c,1)
T_2	(e,1)
T_3	(a,1), **(b,5), (c,1), (d,3)**, (e,1)
T_4	**(b,4), (c,3), (d,3)**, (e,1)
T_5	(a,1), (c,1), (d,1)
T_6	(a,2), (c,6), (e,2)
T_7	(b,2), (c,2), (e,1)

Item	Unit profit
a	5$
b	2$
c	1$
d	2$
e	3$



The utility of the itemset $\{b, c, d\}$ is calculated as follows:

$u(\{b, c, d\}) = (5 \times 2) + (1 \times 1) + (3 \times 2) + (4 \times 2) + (3 \times 1) + (3 \times 2) = 34$

The utility of the generalized itemset $\{Y,c\}$ is calculated as follows:

$u(\{Y, c\}) = (1 \times 5 + 1 \times 1) + (1 \times 5 + 5 \times 2 + 1 \times 1) + (4 \times 2 + 3 \times 1) + (1 \times 5 + 1 \times 1) + (2 \times 5 + 6 \times 1) + (2 \times 2 + 2 \times 1) = 61$

Problem definition

Input:

A transaction database

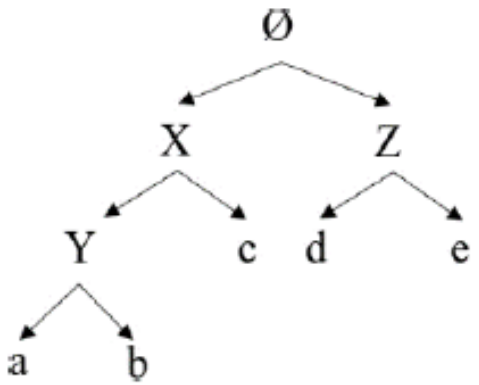
TID	Items
T_1	(a,1), (c,1)
T_2	(e,1)
T_3	(a,1), (b,5), (c,1), (d,3), (e,1)
T_4	(b,4), (c,3), (d,3), (e,1)
T_5	(a,1), (c,1), (d,1)
T_6	(a,2), (c,6), (e,2)
T_7	(b,2), (c,2), (e,1)

External utility values

Item	Unit profit
a	5$
b	2$
c	1$
d	2$
e	3$

a *minutil* threshold

a taxonomy



Output:

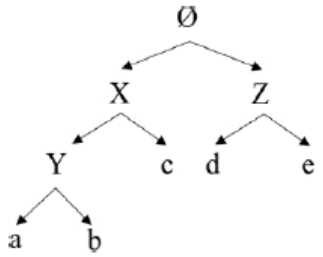
Cross-level high-utility itemsets (with utility≥*minutil*)

if *minutil* = 60$, the *CLHUIs* are:

{X}:61$	{Z,Y,c}:84$
{Y,c}:61$	{e,X}:64$
{Z,X}:84$	{e,Y,c}:64$
{Z,Y}:71$	

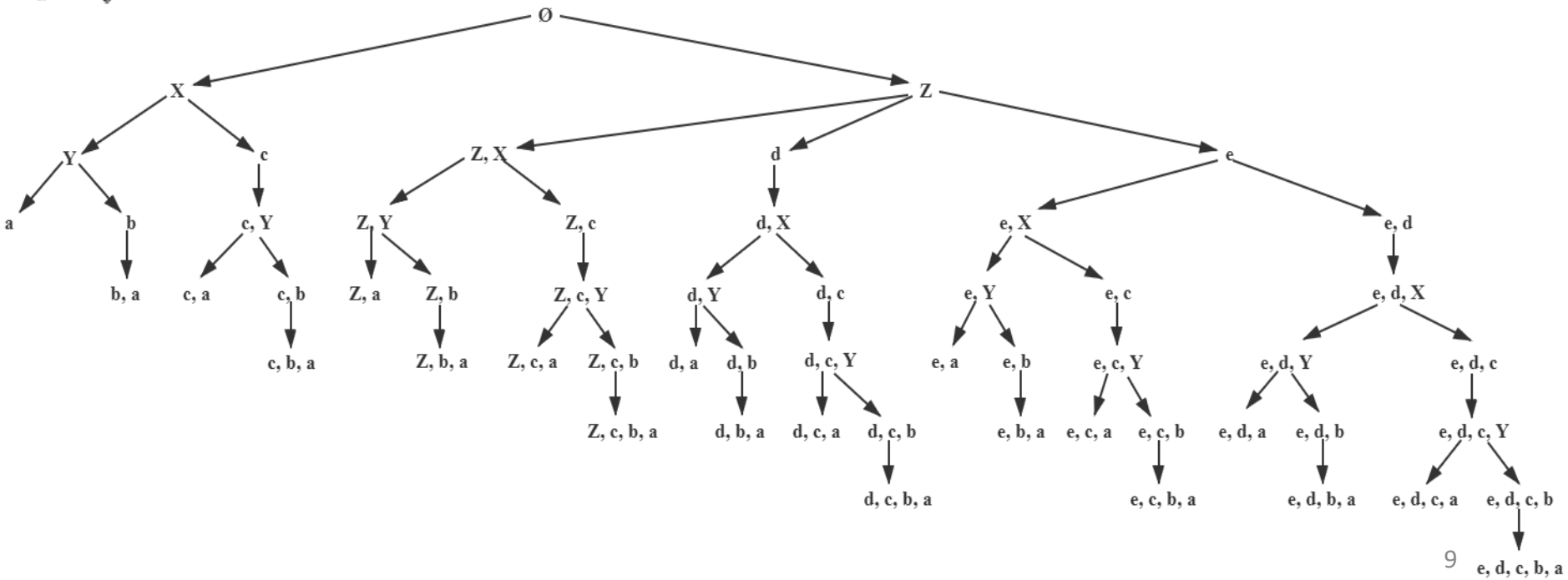
The algorithm

- Based on HUI-Miner, it extends the search space according to two kinds of **extensions**.



Two extensions:

- Tax-based extension
- Join-based extension



The algorithm

- Create a vertical structure named **tax-utility-list** for each itemset.

Tax-utility-list of {Z,Y}

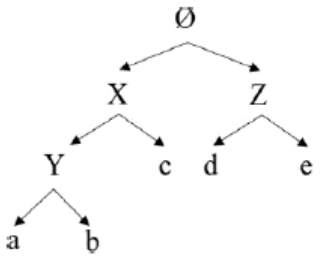
TID	*iutil*	*rutil*
T_3	24	1
T_4	17	3
T_5	7	1
T_6	16	6
T_7	7	2
Childs		
a	b	

utility-list

taxonomy information

utlity of itemset in a transaction

remaining-utility



Construction of a tax-utility-list

- The **tax-utility-list** of a **single item** can be constructed by **scanning the database**.
- For other itemsets, it can be obtained by **joining their child itemset's tax-utility-lists**.

Tax-utility-list of {b}

TID	*iutil*	*rutil*
T_3	10	10
T_4	8	12
T_7	4	5
Childs		

Tax-utility-list of {c}

TID	*iutil*	*rutil*
T_1	1	0
T_3	1	9
T_4	3	9
T_5	1	2
T_6	6	6
T_7	2	3
Childs		

Tax-utility-list of {b,c}

TID	*iutil*	*rutil*
T_3	11	9
T_4	11	9
T_7	6	3
Childs		

How to reduce the search space?

Two properties to improve efficiency:

- **The GWU measure**: $GWU(X) = \sum_{T_c \in g(x)} TU(T_c)$, $where\ TU(T_c) = \sum_{x \in T_c} u(x, T_c)$

  if $GWU(X) < minutil$, prune X and all its extensions.

- **The REU measure**: $REU(X) = \sum_{e \in tuList(X)} (e.iutil + e.rutil)$

  if $REU(X) < minutil$, prune all its tax-based extensions.

Pseudocode

Algorithm 1: The CLH-Miner algorithm

input : D: a transaction database, τ: a taxonomy, $minutil$: the threshold
output: the CLHUIs

1 Scan D and τ to calculate the GWU of each item $i \in I$ and each generalized item $g \in GI$;
2 $I^* \leftarrow \{i | i \in I \wedge GWU(i) \geq minutil \vee \forall g \in GI$ such that $Desc(g, \tau) \ni i$, $GWU(g) \geq minutil\}$;
3 $GI^* \leftarrow \{g | g \in GI \wedge GWU(g) \geq minutil\}$;
4 Calculate the total order $\prec$ on $I^* \cup GI^*$;
5 Scan D and τ to build the $tuList$ of each item $i \in I^*$ and of generalized item $g \in GI^*$;
6 $TULs \leftarrow \{g | g \in GI^* \wedge level(g) = 1\}$;
7 Search $(TULs, minutil)$;

Algorithm 2: The *Search* procedure

input : $TULs$: a set of extensions of an itemset P, $minutil$: the threshold
output: the CLHUIs that are transitive extensions of P

1 **foreach** itemset $X \in TULs$ **do**
2 if $SUM(X.tuList.iutils) \geq minutil$ **then** Output X;
3 $ExtensionsOfX \leftarrow \emptyset$;
4 **foreach** itemset $Y \in TULs$ *such that* $X \prec Y$ **do**
5 $JoinExtension.tuList \leftarrow ConstructJoinExtension(X, Y)$;
6 **if** $GWU(JoinExtension) \geq minutil$ **then**
$ExtensionsOfX \leftarrow ExtensionsOfX \cup \{JoinExtension\}$;
7 **end**
8 if $SUM(X.tuList.iutils) + SUM(X.tuList.rutils) \geq minutil$ **then**
9 **foreach** itemset $T \in X.childs$ **do**
10 $TaxExtension.tuList \leftarrow ConstructTaxExtension(X, T)$;
11 **if** $GWU(TaxExtension) \geq minutil$ **then**
$ExtensionsOfX \leftarrow ExtensionsOfX \cup \{TaxExtension\}$;
12 **end**
13 **end**
14 Search $(ExtensionsOfX, minutil)$;
15 **end**

Experimental evaluation

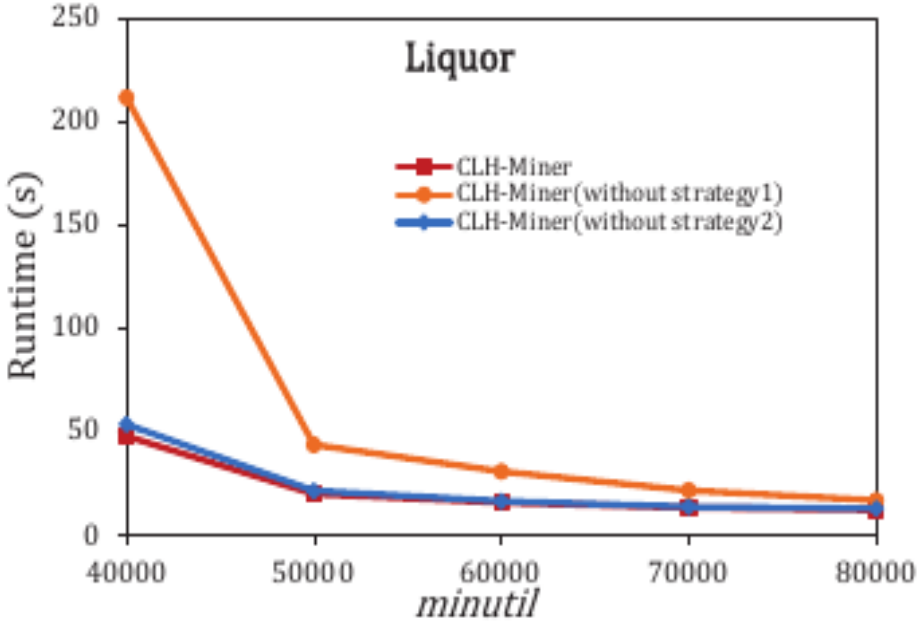
Datasets' characterictics

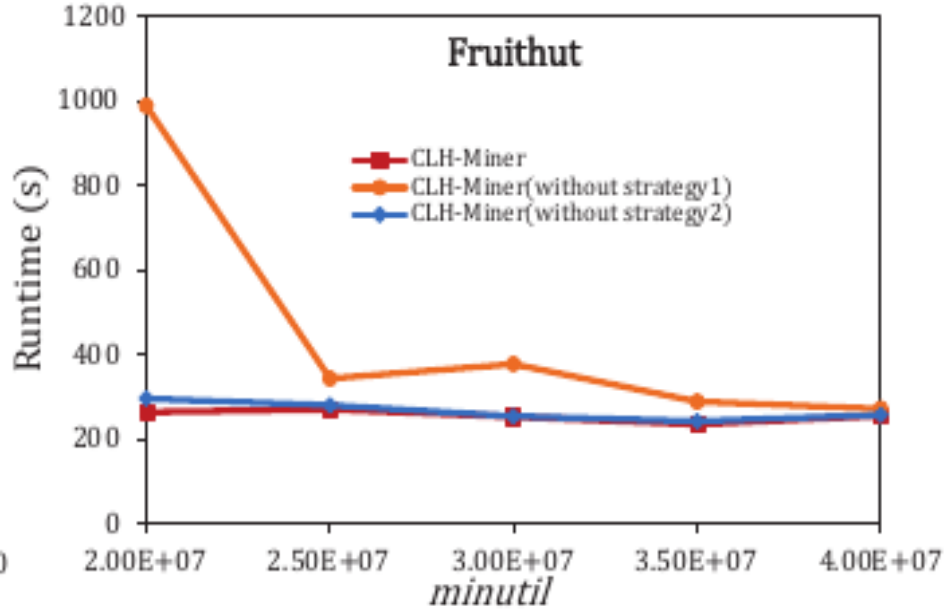
Dataset	$\|D\|$	$\|I\|$	$\|GI\|$	T_{min}	T_{max}	T_{avg}	$Level$
Liquor	9,284	2,626	77	1	5	2.7	7
Fruithut	181,970	1,265	43	23	36	3.58	4

- |D|: number of transactions
- |I|: number of distinct items
- |GI|: number of distinct generalized items
- T_{min}: minimum transaction length
- T_{max}: maximum transaction length
- T_{avg}: average transaction length
- $Level$: the maximum level of items

Both **Liquor** and **Fruithut** are real-life datasets.

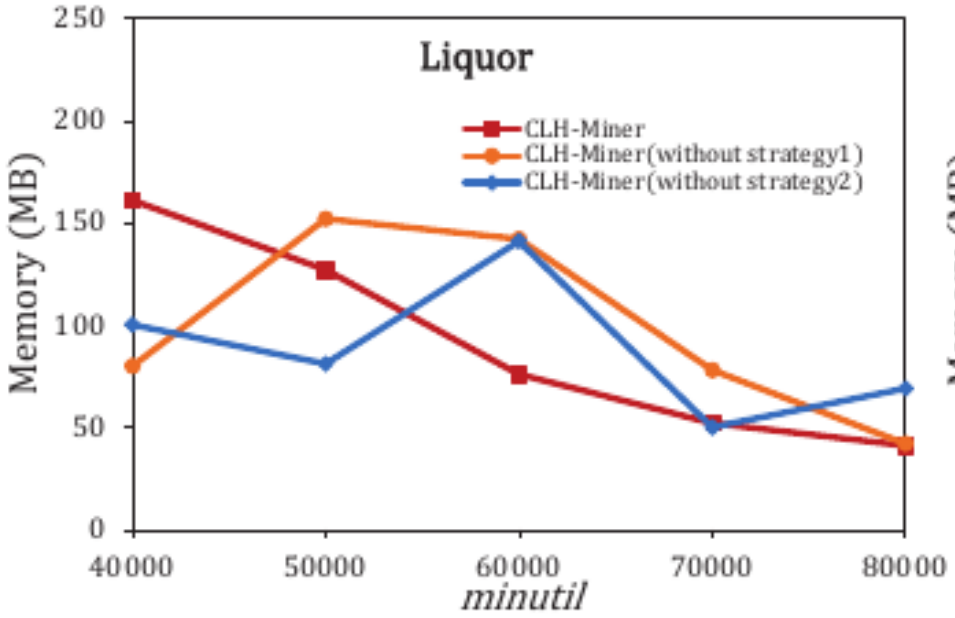
Execution times

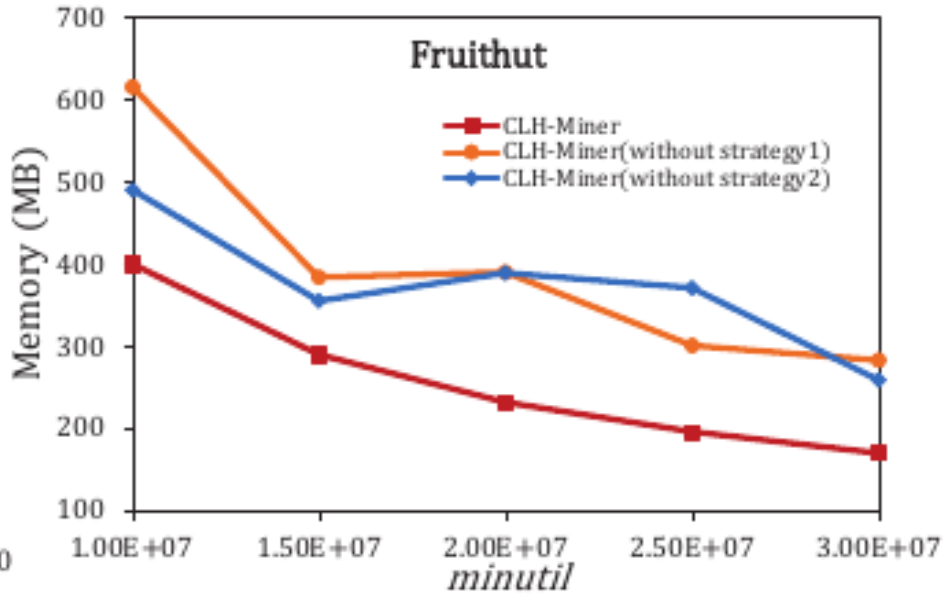




Increasing *minutil* often decrease the runtime.
GWU-based pruning generally greatly decrease the runtime.

Peak memory usage





CLH-Miner consumes less memory for high *minutil* value.

Comparison of CLH-Miner, ML-HUI Miner and HUI-Miner

Table 3. Comparison of CLH-Miner, ML-HUI Miner and HUI-Miner on *Liquor*

minutil	Runtime (s)			Memory (MB)		
	CLH-Miner	ML-HUI Miner	HUI-Miner	CLH-Miner	ML-HUI Miner	HUI-Miner
9k	877.74	39.23	0.08	222	130	59
10k	738.77	40.65	0.08	146	67	104
11k	515.81	39.00	0.06	183	111	27
12k	448.82	39.76	0.05	229	180	73
13k	406.77	40.74	0.05	150	316	118

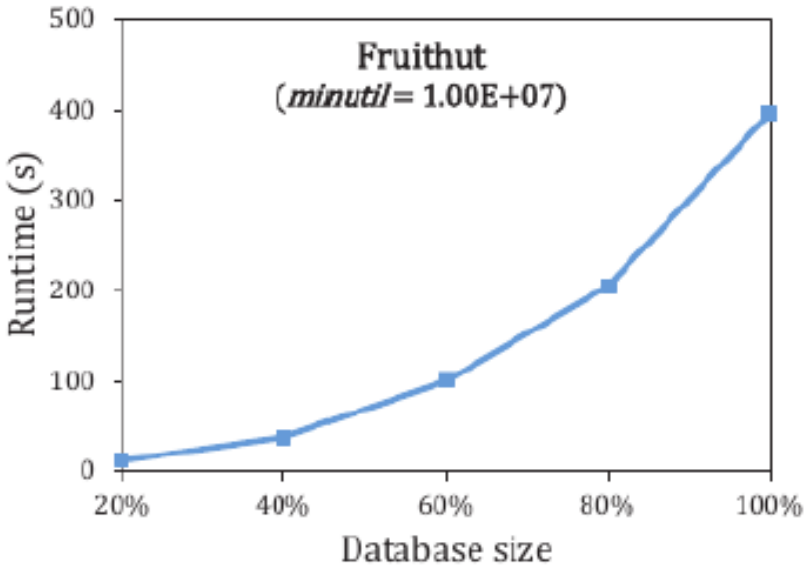
Table 4. Comparison of CLH-Miner, ML-HUI Miner and HUI-Miner on *Fruithut*

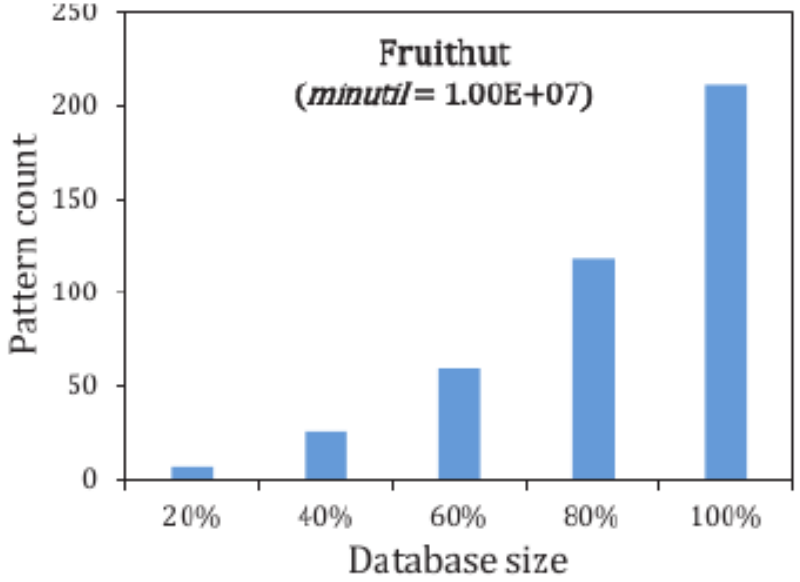
minutil	Runtime (s)			Memory (MB)		
	CLH-Miner	ML-HUI Miner	HUI-Miner	CLH-Miner	ML-HUI Miner	HUI-Miner
4E+06	1,198.20	76.36	1.43	938	378	140
5E+06	1,166.74	83.26	1.16	778	1,788	129
6E+06	806.20	92.22	1.03	668	1,892	121
7E+06	694.89	97.96	0.69	581	2,162	100
8E+06	603.69	96.70	0.55	520	1,803	92

Table 5. Comparison of pattern count on *Liquor* and *Fruithut*

Liquor				*Fruithut*			
minutil	CLHUIs	Multi-level HUIs	HUIs	*minutil*	CLHUIs	Multi-level HUIs	HUIs
10k	5537	234	241	4E+06	1248	55	11
20k	2136	119	86	5E+06	833	45	8
30k	1391	79	43	6E+06	586	36	4
40k	939	56	19	7E+06	435	30	2
50k	469	49	12	8E+06	331	23	1

Scalability





Runtime and pattern count increase with database size.

Conclusion

- Contributions:
  - A new type of patterns: **cross-level high utility itemsets**
  - An efficient algorithm, named **CLH-Miner**
  - Two optimizations: **GWU** and **REU**
- Future work:
  - Consider other types of databases.
  - Adapt to other problems.

Thank you. Questions?

Source code and datasets will be made available as part of the SPMF data mining library