Mining Minimal High-Utility Itemsets

Philippe Fournier-Viger[1](✉), Jerry Chun-Wei Lin[2], Cheng-Wei Wu[3], Vincent S. Tseng[3], and Usef Faghihi[4]

[1] Harbin Institute of Technology Shenzhen, China

[2] National Chiao Tung University, Taiwan

[3] University of Indianapolis, Indianapolis, USA

30 min

High-utility itemset mining

Input

a transaction database

TID	Transaction
T_1	$(a,1),(b,5),(c,1),(d,3),(e,1),(f,5)$
T_2	$(b,4),(c,3),(d,3),(e,1)$
T_3	$(a,1),(c,1),(d,1)$
T_4	$(a,2),(c,6),(e,2),(g,5)$
T_5	$(b,2),(c,2),(e,1),(g,2)$

a unit profit table

Item	a	b	c	d	e	f	g
Profit	5	2	1	2	3	1	1

minutil: *a minimum utility threshold set by the user* (*a positive integer*)

High-utility itemset mining

Input

a transaction database

TID	Transaction
T_1	$(a,1),(b,5),(c,1),(d,3),(e,1),(f,5)$
T_2	$(b,4),(c,3),(d,3),(e,1)$
T_3	$(a,1),(c,1),(d,1)$
T_4	$(a,2),(c,6),(e,2),(g,5)$
T_5	$(b,2),(c,2),(e,1),(g,2)$

a unit profit table

Item	a	b	c	d	e	f	g
Profit	5	2	1	2	3	1	1

***minutil**: a minimum utility threshold set by the user* (*a positive integer*)

Output

All high-utility itemsets (itemsets having a utility $\geq minutil$)

For example, if *minutil = 33$*, the high-utility itemsets are:

{b,d,e} 36$ 2 transactions	{b,c,d} 34$ 2 transactions
{b,c,d,e} 40$ 2 transactions	{b,c,e} 37 $ 3 transactions

Utility calculation

a transaction database

TID	Transaction
T_1	$(a,1),(b,5),(c,1),(d,3),(e,1),(f,5)$
T_2	$(b,4),(c,3),(d,3),(e,1)$
T_3	$(a,1),(c,1),(d,1)$
T_4	$(a,2),(c,6),(e,2),(g,5)$
T_5	$(b,2),(c,2),(e,1),(g,2)$

a unit profit table

Item	a	b	c	d	e	f	g
Profit	5	2	1	2	3	1	1

The **utility** of the itemset {b,d,e} is calculated as follows:

$$u(\{\mathbf{b,d,e}\}) = \underbrace{(5\text{x}2)+(3\text{x}2)+(3\text{x}1)}_{\text{utility in transaction } T_1} + \underbrace{(4\text{x}2)+(2\text{x}3)+(1\text{x}3)}_{\text{utility in transaction } T_2} = \mathbf{36\$}$$

Challenge in Utility Mining

a transaction database

TID	Transaction
T_1	$(a,1),(b,5),(c,1),(d,3),(e,1),(f,5)$
T_2	$(b,4),(c,3),(d,3),(e,1)$
T_3	$(a,1),(c,1),(d,1)$
T_4	$(a,2),(c,6),(e,2),(g,5)$
T_5	$(b,2),(c,2),(e,1),(g,2)$

a unit profit table

Item	a	b	c	d	e	f	g
Profit	5	2	1	2	3	1	1

The utility measure is not monotonic or anti-monotonic

u({a,b,c,d,e}) = 25 $

u({b,d,e}) = 36 $

u({b,d}) = 30 $

The utility of an itemset can be lower, greater or equal to the utility of its subsets.

Previous work uses monotonic upper-bounds to reduce the search space.

Problem

High-utility itemset mining

- is **useful** for discovering **profitable itemsets**.
- but it can find **a large amount of itemsets**
- **some itemsets are redundant**
- **solution:**
  - discover **concise representations of HUIs**
  - Previous work →

Concise representations

- **Maximal HUIs**: HUIs not included in another HUI.
  - GUIDE
- **Closed HUIs:** HUIs having no supersets appearing in the same transactions
  - CHUI-Miner, CHUD…
- **Generators of HUIs:** smallest set of itemsets common to a set of transactions containing a HUI
  - GHUI-Miner…

Limitations of previous work

- Other algorithms often find many long itemsets
- It is easier to co-promote a small set of items targeted at many customers rather than a large set of items targeted at few customers.
- **Proposal**:
  - **Minimal HUIs**: smallest sets of items that yield a high profit.
  - Not considered in previous work.

What is a minimal high utility itemset?

A **MinHUI** is a **high-utility itemset** that has no proper subset that is a high-utility itemset.

For example:

{b,d,e} 36$ 2 transactions	{b,c,d} 34$ 2 transactions
~~{b,c,d,e} 40$~~ ~~2 transactions~~	{b,c,e} 37 $ 3 transactions

Interesting properties

Property 2

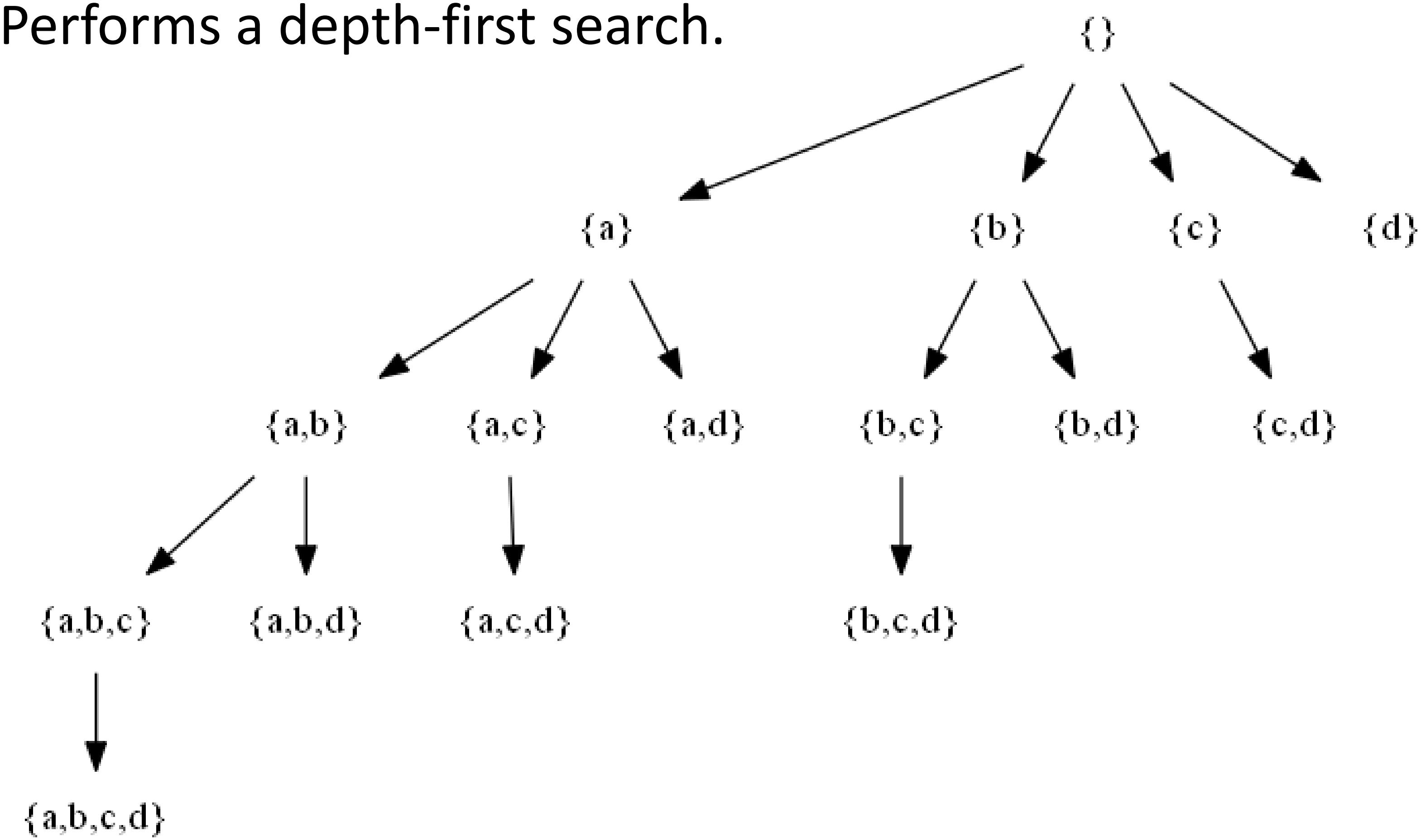
- If an itemset X is a MinHUI, then its subsets and supersets are not MinHUIs.

Property 1

- If *minutil* is lowered, the number of MinHUIs may increase, decrease or stay the same.
- If *minutil* is set to 0, the number of MinHUIs is equal to the number of items.

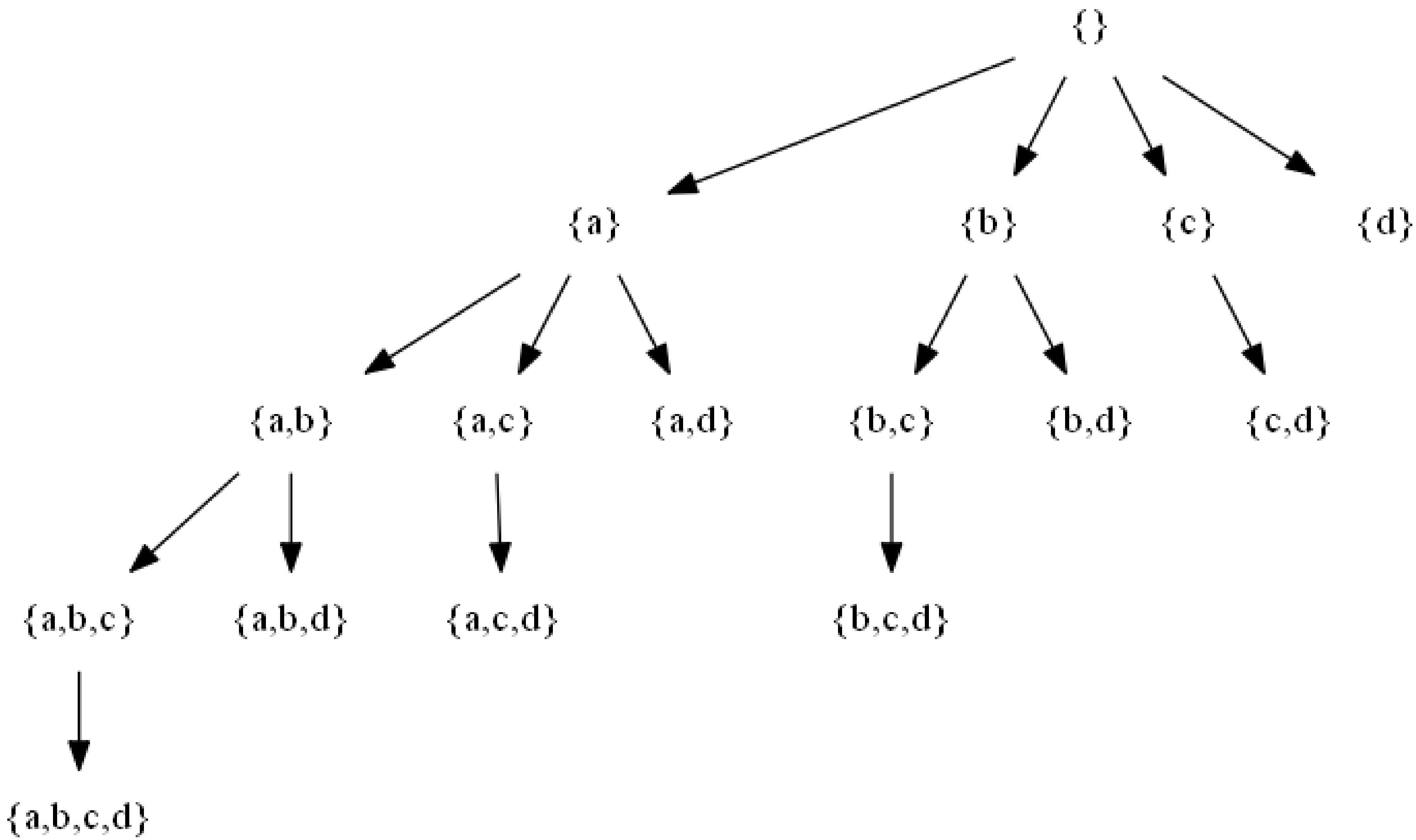
The MinFHM algorithm

- An algorithm for mining minimal high utility-itemsets
- Extends FHM
- Performs a depth-first search.



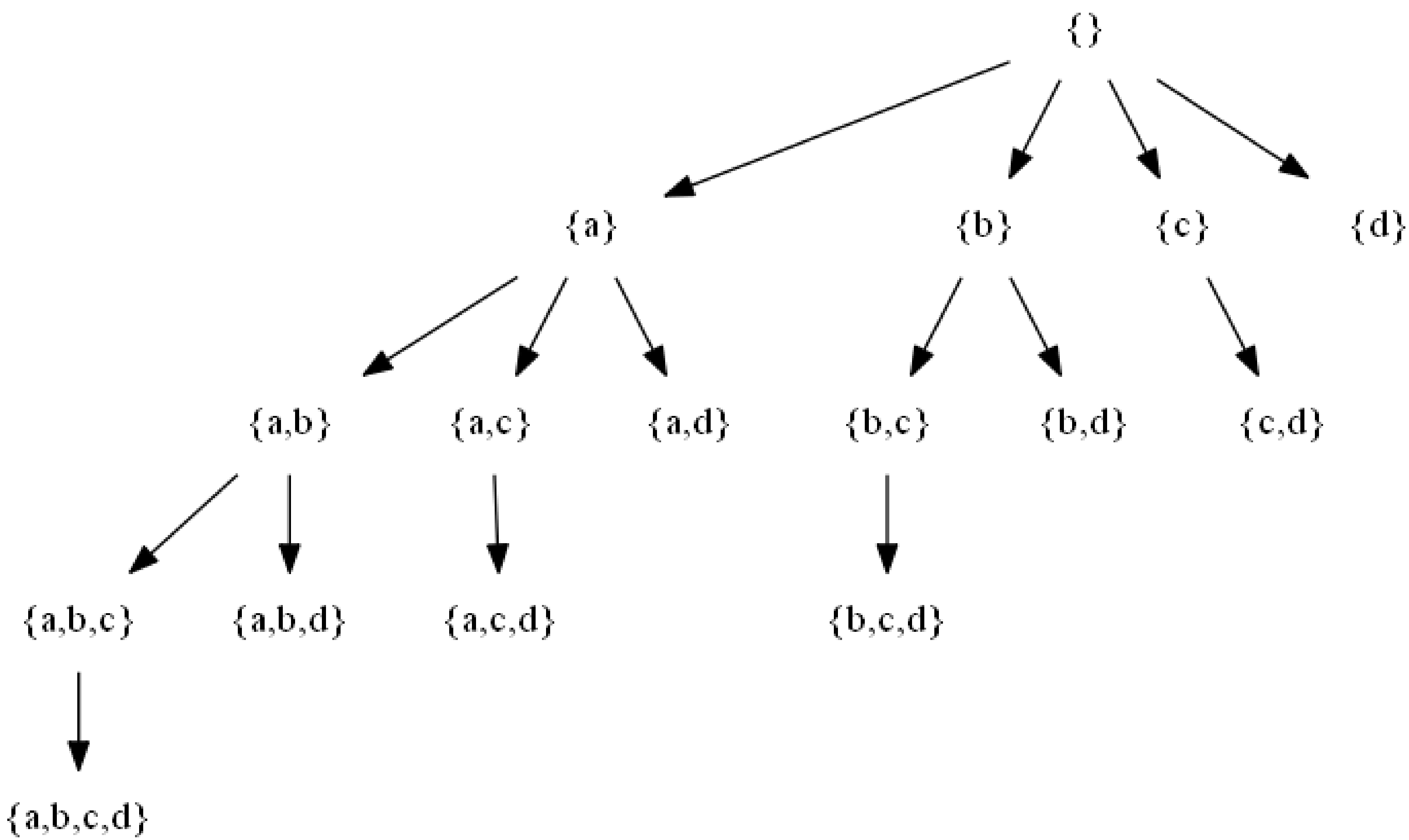
The MinFHM algorithm

- Applies Property 2 to prune the search space.
- If an itemset is a MinHUI, its supsersets are not MinHUIs



The MinFHM algorithm

Calculates an upper bound on the utility of extensions of each itemset to decide whether its extensions should be explored.



Creating utility-lists of items

The algorithm scans the database to create a *utility-list* structure for each item

Itemset {a}

TID	Utility	Remaining Utility
T1	5	20
T3	5	3
T4	10	12

Itemset {d}

TID	Utility	Remaining Utility
T1	6	3
T2	6	3
T3	2	0

Calculating their utility

The utility of each item is calculated to determine if it is a high utility itemset

Itemset {a}

TID	Utility	Remaining Utility
T1	5	20
T3	5	3
T4	10	12

Utility: **20$**

Itemset {d}

TID	Utility	Remaining Utility
T1	6	3
T2	6	3
T3	2	0

Utility: **14$**

Calculating their upper-bounds

An upper-bound is calculated on the utility of extensions of each item.

Itemset {a}

TID	Utility	Remaining Utility
T1	5	20
T3	5	3
T4	10	12

$\underbrace{\textbf{20\$} \quad \textbf{35\$}}$

Upper-bound: **55$**

Itemset {d}

TID	Utility	Remaining Utility
T1	6	3
T2	6	3
T3	2	0

This indicates that extensions of {a} cannot have a utility higher than 55$

Generating a larger itemset

Itemset {a}

TID	Utility	Remaining Utility
T1	5	20
T3	5	3
T4	10	12

Itemset {d}

TID	Utility	Remaining Utility
T1	6	3
T2	6	3
T3	2	0

Itemset {a,d}

TID	Utility	Remaining Utility
T1	11	3
T3	7	0

Utility-lists of larger itemsets are generated by joining the utility-lists of some of its subsets. No need to scan the database!

Calculating its utility

Itemset {a}

TID	Utility	Remaining Utility
T1	5	20
T3	5	3
T4	10	12

Itemset {d}

TID	Utility	Remaining Utility
T1	6	3
T2	6	3
T3	2	0

Itemset {a,d}

TID	Utility	Remaining Utility
T1	11	3
T3	7	0

Utility: **18$**

Calculating its upper-bound

Itemset {a}

TID	Utility	Remaining Utility
T1	5	20
T3	5	3
T4	10	12

Itemset {d}

TID	Utility	Remaining Utility
T1	6	3
T2	6	3
T3	2	0

Itemset {a,d}

TID	Utility	Remaining Utility
T1	11	3
T2	7	0
	18$	**3$**

Upper-bound: **21$**

Extensions cannot have a utility greater than 21$

Pseudocode

Algorithm 1. The MinFHM algorithm

input : D: a transaction database, $minutil$: a user-specified threshold
output: the set of high-utility itemsets

Scan D to calculate the TWU of single items;
$I^* \leftarrow$ each item i such that $\text{TWU}(i) \geq minutil$;
Let $\succ$ be the total order of TWU ascending values on I^*;
Scan D to build the utility-list of each item $i \in I^*$ and build the $EUCS$;
Output each item $i \in I^*$ such that $\text{SUM}(\{i\}.utilitylist.iutils) \geq minutil$;
`Search` $(\emptyset, I^*, minutil, EUCS)$;

Algorithm 2. The *Search* procedure

input : P: an itemset, $ExtensionsOfP$: a set of extensions of P, , $minutil$: a user-specified threshold, $EUCS$: the $EUCS$
output: the set of high-utility itemsets

foreach *itemset* $Px \in ExtensionsOfP$ **do**
    if $SUM(Px.utilitylist.iutils)+SUM(Px.utilitylist.rutils) \geq minutil$ **then**
        $ExtensionsOfPx \leftarrow \emptyset$;
        foreach *itemset* $Py \in ExtensionsOfP$ *such that* $y \succ x$ **do**
            if $\exists(x, y, c) \in EUCS$ *such that* $c \geq minutil)$ **then**
                $Pxy \leftarrow Px \cup Py$;
                $Pxy.utilitylist \leftarrow$ `Construct` (P, Px, Py);
                $ExtensionsOfPx \leftarrow ExtensionsOfPx \cup \{Pxy\}$;
                if $SUM(Pxy.utilitylist.iutils) \geq minutil$ **then** output Px;
            end
        end
        `Search` $(Px, ExtensionsOfPx, minutil)$;
    end
end

Algorithm 3. The Construct procedure

input : P: an itemset, Px: the extension of P with an item x, Py: the extension of P with an item y
output: the utility-list of Pxy

$UtilityListOfPxy \leftarrow \emptyset$;
foreach *tuple* $ex \in Px.utilitylist$ **do**
    if $\exists ey \in Py.utilitylist$ *and* $ex.tid = exy.tid$ **then**
        if $P.utilitylist \neq \emptyset$ **then**
            Search element $e \in P.utilitylist$ such that $e.tid = ex.tid$.;
            $exy \leftarrow (ex.tid, ex.iutil + ey.iutil - e.iutil, ey.rutil)$;
        end
        else
            $exy \leftarrow (ex.tid, ex.iutil + ey.iutil, ey.rutil)$;
        end
        $UtilityListOfPxy \leftarrow UtilityListOfPxy \cup \{exy\}$;
    end
end
return $UtilityListPxy$;

Some details have not been explained in the presentation. See the paper for more details.

Experimental Evaluation

Datasets’ characterictics

Dataset	transaction count	distinct item count	average transaction length
Mushroom	8,124	119	23.0
Kosarak	990,000	41,270	8.1
Retail	16,470	88,162	10.3
Foodmart	4,141	1,559	4.14

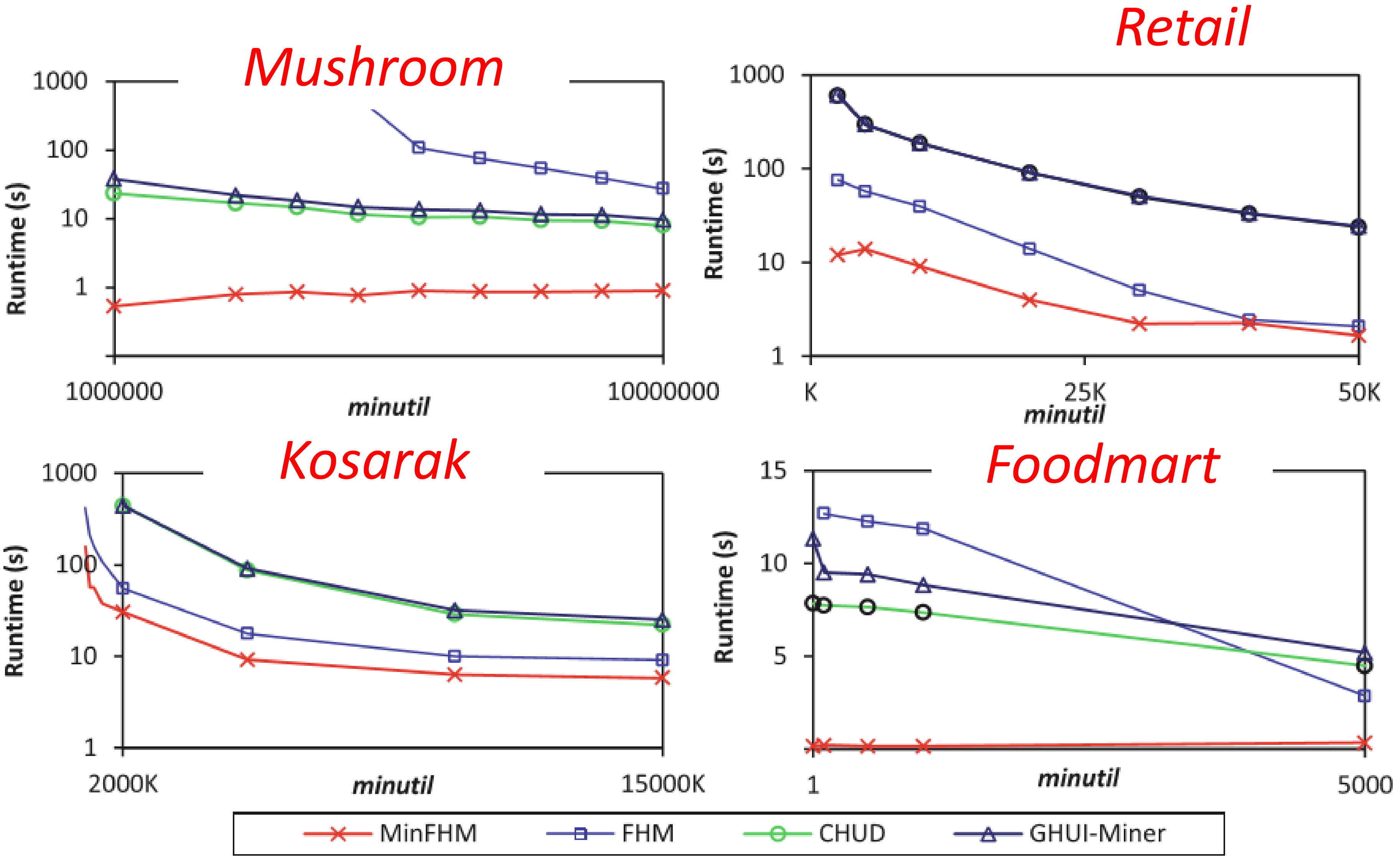
- **Foodmart** is a real-life transaction datasets from retail stores.
- External and internal utility values have been generated in the [1, 000] and [1, 5] intervals using a log-normal distribution

Experimental Evaluation

- We compared the performance of MinFHM with
  - FHM for mining HUIs
  - CHUD for mining closed HUIs
  - GHUI-Miner for mining generators of HUIs

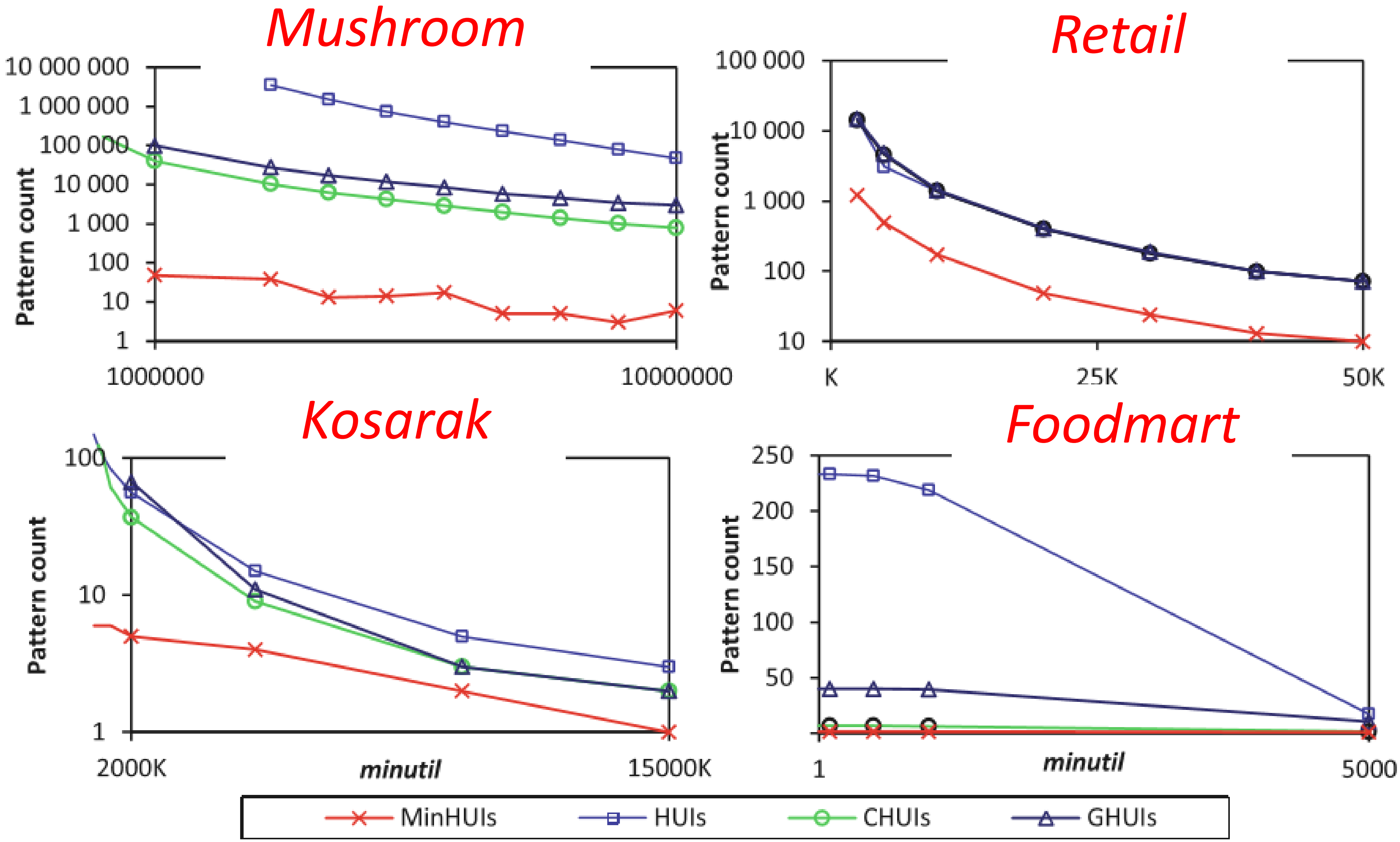
- We varied the *minutil* threshold and compared execution time, memory usage, number of patterns
- Computer with 12 GB of RAM, Java, Windows 7, 64 bit Core i5 Processor

Execution times



MinFHM is up to 800 times faster

Number of patterns



MinFHM is up to 900,000 times less patterns

Conclusion

- Contribution:
  - New task : mining **minimal high utility itemsets**
  - Properties and an algorithm: **MinFHM**
- Experimental results:
  - up to **800 times faster**
  - Very compact: up to **900,000 times less patterns** than other concise representations of HUIs
- Source code and datasets available as part of the **SPMF data mining library (**GPL 3).

SPMF

Open source Java data mining software, 120 algorithms
http://www.phillippe-fournier-viger.com/spmf/

Thank you. Questions?

Open source Java data mining software, 120 algorithms
http://www.phillippe-fournier-viger.com/spmf/