Mining Local High Utility Itemsets

Philippe Fournier-Viger[1], Yimin Zhang[2], Jerry Chun-Wei Lin[3], Yun Sing Koh[4], and Hamido Fujita[5]

[1] School of Natural Sciences and Humanities, Harbin Institute of Technology, Shenzhen, China
[2] School of Computer Sciences and Technology, Harbin Institute of Technology, Shenzhen, China
[3] Department of Computing, Mathematics and Physics, Western Norway University of Applied Sciences (HVL), Bergen, Norway
[4] Department of Computer Sciences, University of Auckland, Auckland, New Zealand
[5] Faculty of Software and Information Science, Iwate Prefectural University, Iwate, Japan

High Utility Itemset (HUI) Mining

- **Input:**

A transaction database **a unit profit table** **a minimum threshold** $minutil$

internal utility

TID	Items
T_1	b(2),c(2),e(1)
T_2	b(4),c(3),d(2),e(1)
T_3	b(2),c(2),e(1)
T_4	a(2),b(10),c(2),d(10),e(2)
T_5	a(2),c(6),e(2)
T_6	b(4),c(3),e(1)
T_7	a(2),c(2),d(2)
T_8	a(2),c(6),e(2)

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

external utility

For example, if $minutil = 60\$$, **the** $HUIs$ **are**

$\{b, e\}: 62\$$	$\{a, c, e\}: 62\$$
$\{b, d, e\}: 78\$$	$\{b, c, d, e\}: 85\$$
$\{b, c, e\}: 74\$$	

- **output:**

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

Utility Calculation

TID	Items
T_1	b(2),c(2),e(1)
T_2	b(4),c(3),d(2),e(1)
T_3	b(2),c(2),e(1)
T_4	a(2),b(10),c(2),d(10),e(2)
T_5	a(2),c(6),e(2)
T_6	b(4),c(3),e(1)
T_7	a(2),c(2),d(2)
T_8	a(2),c(6),e(2)

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:

$$u(\{b,c,d\}) = \underbrace{(4 \times 2) + (3 \times 1) + (2 \times 2)}_{\text{utility in transaction } T_2} + \underbrace{(10 \times 2) + (2 \times 1) + (10 \times 2)}_{\text{utility in transaction } T_4} = 57$$

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 monotonic to be able to prune the search space.**

Limitation

- High utility itemset mining
  - ➢is **useful** for discovering profitable itemsets in the **whole database**
  - ➢but ignores the time **when** transactions were made
  - ➢fails to find itemsets that have a high utility in **some time periods**

- we proposed a new type of pattern: **local high utility itmeset, LHUI**
  - ➢e.g. $\{schoolbag, pen, notebook\}$ yields a high profit during the back-to-school shopping season, while not being a HUI in the whole year.

Database and Concepts

TID	Items	timestamp
T_1	b(2),c(2),e(1)	d_1
T_2	b(4),c(3),d(2),e(1)	d_3
T_3	b(2),c(2),e(1)	d_3
T_4	a(2),b(10),c(2),d(10),e(2)	d_5
T_5	a(2),c(6),e(2)	d_6
T_6	b(4),c(3),e(1)	d_7
T_7	a(2),c(2),d(2)	d_9
T_8	a(2),c(6),e(2)	d_{10}

$$W_{d_1,d_5} = \{T_1, T_2, T_3, T_4\}$$

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

Timestamps of transactions $T_1, T_2 \ldots T_8$ are $d_1, d_3, \ldots d_{10}$, representing days ($d_i = ith$ day), note that transactions can be simultaneous.

A window denoted as $W_{i,j}$ is the set of transactions from time i to j, i.e. $W_{i,j} = \{T | i \leq t(T) \leq j\}$

Problem Difinition

- **An itemset X is a local high utility itemset (LHUI) if there exists a window $W_{i,j}$ such that $length(W_{i,j}) = minLength$ and $u_{i,j}(X) > lMinutil$**

TID	Items	timestamp
T_1	b(4),c(2),e(3)	d_1
T_2	b(8),c(3),d(4),e(3)	d_3
T_3	b(4),c(2),e(3)	d_3
T_4	a(10),b(20),c(2),d(20),e(6)	d_5

Theorem. If $minutil = lMinutil \times \lceil W_D / minLength \rceil$, then $HUIs \subseteq LHUIs$.

$$u_{d_1,d_3}(\{b,c\}) = 6 + 11 + 6 = 23 > 20$$

e.g. for $minLength = 3, lMinutil = 20$, then $\{b,c\}$ is a LHUI

Problem Definition

For an itemset X, a window $W_{i,j}$ is a LHUI period if for each window $W_{k,l} \subseteq W_{i,j}$ of length $minLength$, $u_{k,l}(X) \geq lMinutil$.

e.g. for $minLength = 3$, $lMinutil = 20$, W_{d_3,d_7} is a LHUI period of $\{b, c\}$

TID	Items	timestamp
T_1	b(2),c(2),e(1)	d_1
T_2	b(4),c(3),d(2),e(1)	d_3
T_3	b(2),c(2),e(1)	d_3
T_4	a(2),b(10),c(2),d(10),e(2)	d_5
T_5	a(2),c(6),e(2)	d_6
T_6	b(4),c(3),e(1)	d_7
T_7	a(2),c(2),d(2)	d_9
T_8	a(2),c(6),e(2)	d_{10}

W_{d_3,d_7}

$u_{d_3,d_5}(\{b, c\}) = 39 > 20$

$u_{d_4,d_6}(\{b, c\}) = 22 > 20$

$u_{d_5,d_7}(\{b, c\}) = 33 > 20$

Problem Definition

- **The problem of Local High Utility Itemset Mining is to find all LHUIs and their maximum LHUI periods given parameters** $minLength$ **and** $lMinutil$**.**

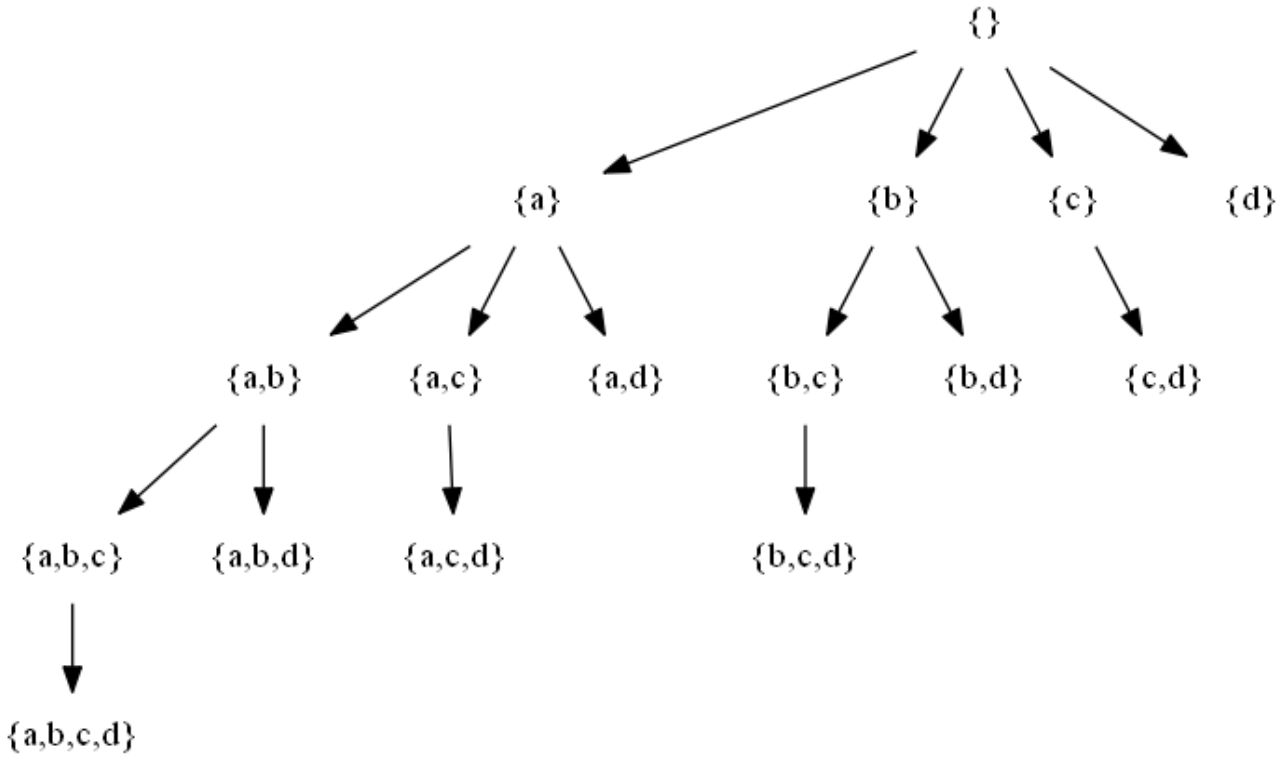
For example, given the database as mentioned before, and set the parameters $minLength = 5$**,** $lMinutil = 30$ **for LHUI and** $minutil = 60$ **for HUI**

25 LHUIs can be found

$\{a, b, c\}: [d_5, d_5]$	$\{\boldsymbol{a}, \boldsymbol{c}, \boldsymbol{e}\}: [d_5, d_{10}]$	$\{a, d\}: [d_5, d_9]$
$\{\boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d}, \boldsymbol{e}\}: [d_3, d_5]$	$\{\boldsymbol{b}, \boldsymbol{c}, \boldsymbol{e}\}: [d_1, d_7]$	$\{\boldsymbol{b}, \boldsymbol{d}, \boldsymbol{e}\}: [d_3, d_5]$
$\{\boldsymbol{b}, \boldsymbol{e}\}: [d_1, d_7]$	$\{c, e\}: [d_3, d_7]$	…

The LHUI-Miner Algorithm

- **Based on HUI-Miner, find larger itemsets with depth-first search;**
- **Create a vertical structure named LU-list for each itemset;**



LU-list of $\{b\}$

utility of itemset in transacti

remaining utility

Utility-list		
tid	$iutil$	$rutil$
T_1	4	5
T_2	8	10
T_3	4	15
T_4	20	28
T_6	8	6
$iutilPeriods$		
$[d_1, d_3], [d_3, d_7]$		
$utilPeriods$		
$[d_1, d_7]$		

the LHUI periods

promising LHUI period

Construction of LU-list

The Utility-list of a single item can be constructed by scanning the database, and others can be obtained by joining their child itemset's Utility-lists

Utility-list $\{b\}$

tid	$iutil$	$rutil$
T_1	4	5
T_2	8	10
T_3	4	15
T_4	20	28
T_6	8	6

Utility-list $\{c\}$

tid	$iutil$	$rutil$
T_1	2	3
T_2	3	7
T_3	2	13
T_4	2	26
T_5	6	6
T_6	3	3
T_7	2	4
T_8	6	6

Utility-list $\{b, c\}$

tid	$iutil$	$rutil$
T_1	6	3
T_2	11	7
T_3	7	13
T_4	22	26
T_6	11	3

Construction of LU-list

- **Consider that** $a \prec b \prec c \prec d \prec e$, $minLength = 3$ **and** $lMinutil = 20$,
- **using sliding window to get periods information**

Utility-list $\{b\}$		
tid	$iutil$	$rutil$
T_1	4	5
T_2	8	10
T_3	4	15
T_4	20	28
T_6	8	6

$sumIutil = 20$
$sumRutil = 30$
$sumUtil = 50$

$sumIutil = 12$
$sumRutil = 20$
$sumUtil = 32$

$sumIutil = 32$
$sumRutil = 53$
$sumUtil = 85$

$sumIutil = 20$
$sumRutil = 28$
$sumUtil = 58$

$sumIutil = 28$
$sumRutil = 34$
$sumUtil = 62$

$iutilPeriods$
$[d_1, d_3], [d_3, d_7]$
$utilPeriods$
$[d_1, d_7]$

T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8
d_1	d_3	d_3	d_5	d_6	d_7	d_9	d_{10}

Optimizations

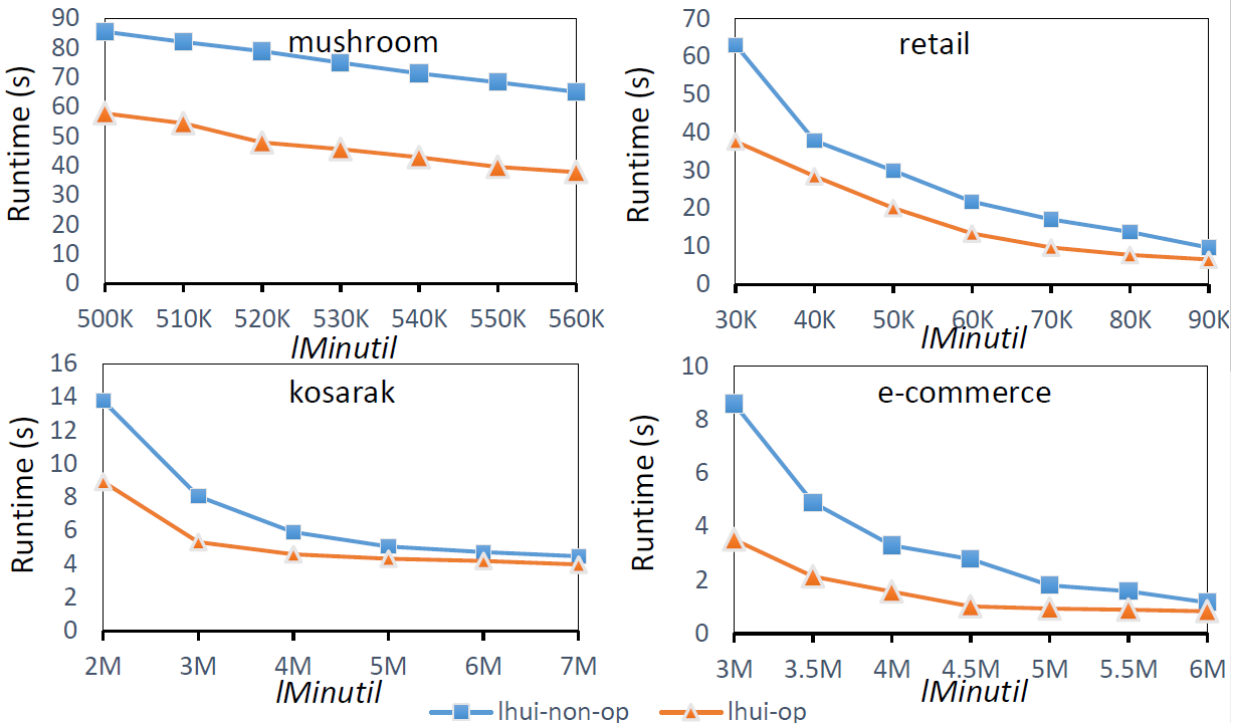
- **Discarding unpromising items using the sliding window;**
  - **Discard an item i, if for any window $W_{k,l}$ of $minLength$, $TWU_{k,l}(i) < lMinUtil$**
- **Discarding irrelevant transactions**
  - **remove transactions that don't contribute to any LHUI**
- **Discarding unpromising tuples in LU-list**
  - **only keep tuples that are in $utilperiods$**

Experimental Evaluation

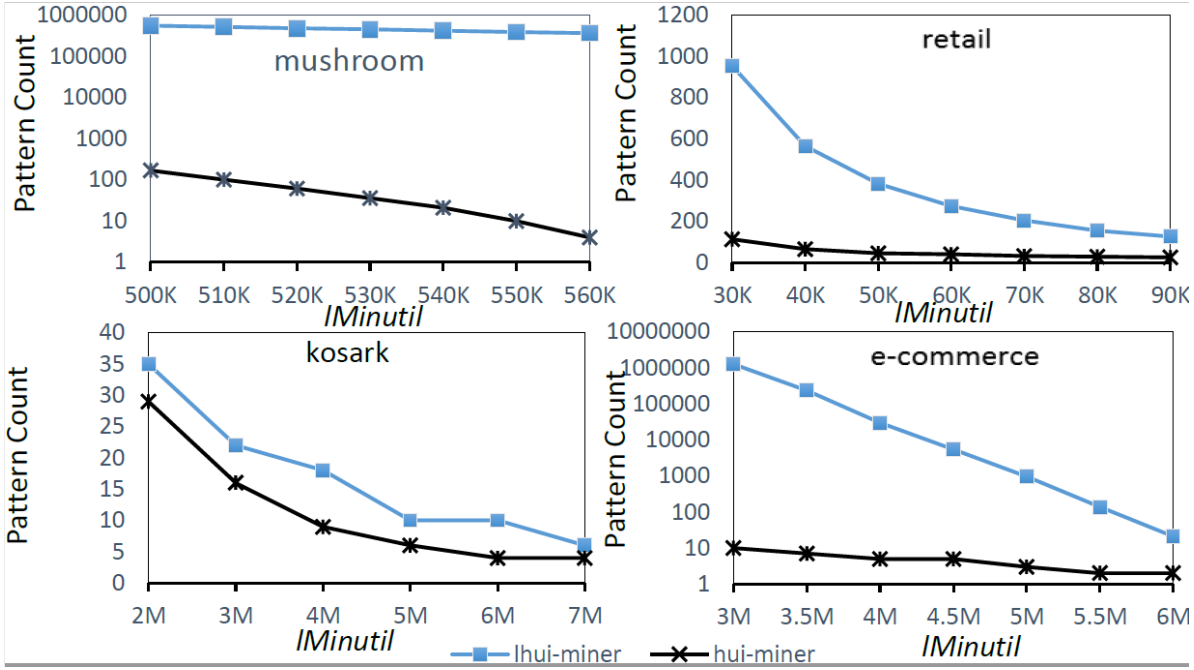
Dataset	Trans count	Item count	Average length	Type
mushroom	8,124	119	23	dense
kosarak	990,000	41,270	8.09	Long transaction
retail	88,162	16,470	10.3	sparse
e_commerce	17,535	3,803	15.4	Real-life data

- **We compared the execution time of the algorithm with and without the optimizations**
- **We also compared the number of patterns found (LHUI and HUI)**
- **java, Windows 10, 16 GB RAM, Intel Xeon E3-1270 v5**

Experimental Evaluation



In some cases, the optimized algorithm is one time faster than the non-optimized algorithm



The number of LHUIs is much more than the number of HUIs in most cases.

An example: for $lMinutil = 1{,}432{,}360$ and $minlength = 90\ days$, the itemset $\{pink\ polkadot\ bag, black\ and\ white\ baroque\ bag\}$ has a LHUI period from 2011/07/19 to 2011/11/02 where it generates a high utility, while the itemset is not a HUI in the whole database for minutil = 6,000,000.

Conclusion

- A new type of patterns named **Local High Utility Itemset;**
- A new algorithm with three optimizations;
- **Results:**
  - the optimizations can reduce running time by half in some cases;
  - the number of LHUI is far more than that of HUI in most cases

- **Perspectives:**
  - adapt the **local** concept to other pattern mining problems;
  - use concise representations of LHUIs

Thank you. Questions?