Discovering High Utility Change Points in Customer Transaction Data

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

[1] School of Natural Sciences and Humanities, Harbin Institute of Technology (Shenzhen), Shenzhen, China
[2] School of Computer Sciences and Technology, Harbin Institute of Technology (Shenzhen), 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

High Utility Itemset Mining

Input:

A transaction database

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)

a unit profit table

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

a $minutil$ threshold

Output:

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

if $minutil = 60\$$, the $HUIs$ are:

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

How to calculate the 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$

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 a **whole database**
  - ➢but it ignores the time **when** transactions were made
  - ➢and it fails to capture **changes in the utility of itemsets** over time

- We propose a new pattern type:

  High Utility Change Point (HUCP)

  e.g.$\{moon\ cake\}$ has upward and downward selling trends before and after the Mid-Autumn Festival, respectively.

Database with time, Window

A transaction database with time

TID	Items	Time
T_1	b(2),c(2),e(1)	d1
T_2	b(4),c(3),d(2),e(1)	d3
T_3	b(2),c(2),e(1)	d3
T_4	a(2),b(10),c(2),d(10),e(2)	d5
T_5	a(2),c(6),e(2)	d6
T_6	b(4),c(3),e(1)	d7
T_7	a(2),c(2),d(2)	d9
T_8	a(2),c(6),e(2)	d10

$$W_{1,5} = \{T_1, T_2, T_3, T_4\}$$

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

- Transactions $T_1, T_2 \ldots T_8$ have **timestamps** $d_1, d_3, \ldots d_{10}$.
- Transactions can be simultaneous.
- A **window** denoted as $W_{i,j}$ is the set of transactions from time i to j, i.e. $W_{i,j} = \{T | i \leq t(T) \leq j\}$

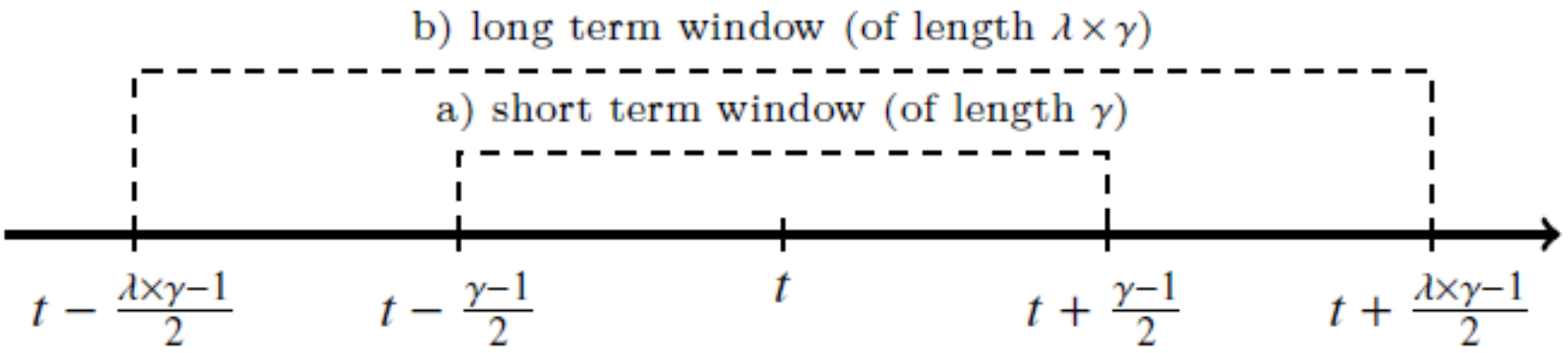
How to detect changes?

Inspiration:

- stock market analysis,
- the crossover of a short-term and a long-term moving average is a buying or selling signal.

MA= Moving Average

Problem Definition



Moving average utility (mau):
the average utility of X before and after t in a window of length λ

Change point: the crossover of long-term and short-term moving average utility line

Problem Definition

- Change points of which itemsets?
- **Local high utility itemset (LHUI):**
  An itemset X is a LHUI if there exists a window $W_{i,j}$ such that $length(W_{i,j}) = winLength$ and $u_{i,j}(X) > lMinutil = minMau \times winLength$

TID	Items	time
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)	d_5

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

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

$minMau$ is a user-defined parameter

Problem Definition

Goal: discover all LHUIs and their change points some parameters $winLength$, $minMau$ and λ.

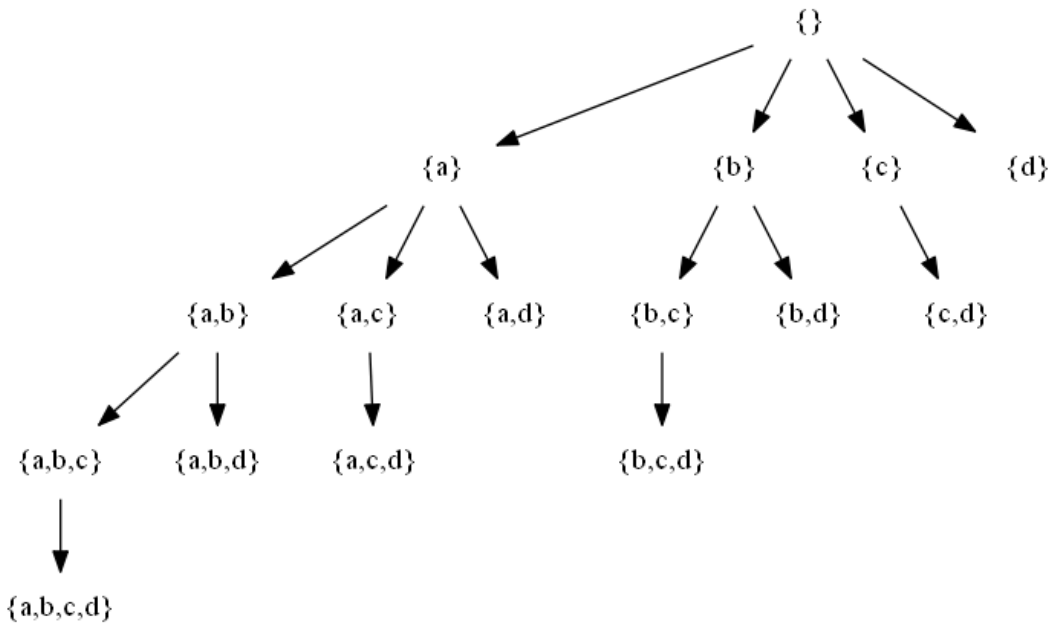
Example:

- Assume that $winLength = 3$, $minMau = 10$ and $\lambda = 1.67$,
- **24 LHUIs** are found with their change points, including: $\{d\}$: $\{d_3, d_6\}$, $\{a, c\}$: $\{d_5, d_7, d_9, d_{10}\}$

TID	Items	Time
T_1	b(2),c(2),e(1)	d1
T_2	b(4),c(3),**d(2)**,e(1)	**d3**
T_3	b(2),c(2),e(1)	d3
T_4	**a(2)**,b(10),**c(2)**,**d(10)**,e(2)	**d5**
T_5	**a(2)**,**c(6)**,e(2)	**d6**
T_6	b(4),c(3),e(1)	**d7**
T_7	**a(2)**,**c(2)**,d(2)	**d9**
T_8	**a(2)**,**c(6)**,e(2)	**d10**

The HUCP-Miner Algorithm

- Based on **LHUI-Miner**, it finds larger itemsets using a **depth-first search**;
- Create a vertical structure named **LU-List** for each item.



LU-list of $\{b\}$

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

utility of itemset in a transaction

remaining utility

LHUI periods

promising LHUI periods

Construction of a Utility-list

- The **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 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 $minMau = 20$,
- using two sliding windows to get long-term and short-term mau

Utility-list $\{b\}$			
tid	$iutil$	$rutil$	Time
T_1	4	5	d1
T_2	8	10	d3
T_3	4	15	d3
T_4	20	28	d5
T_6	8	6	d7

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

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

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

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

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

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

Three 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 $utilperiods$

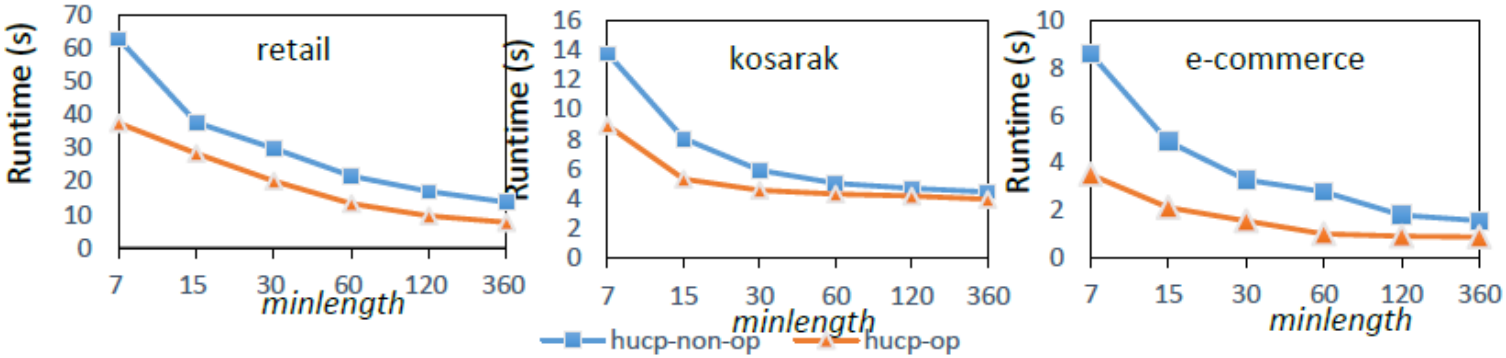
Experimental Evaluation

- Three datasets:

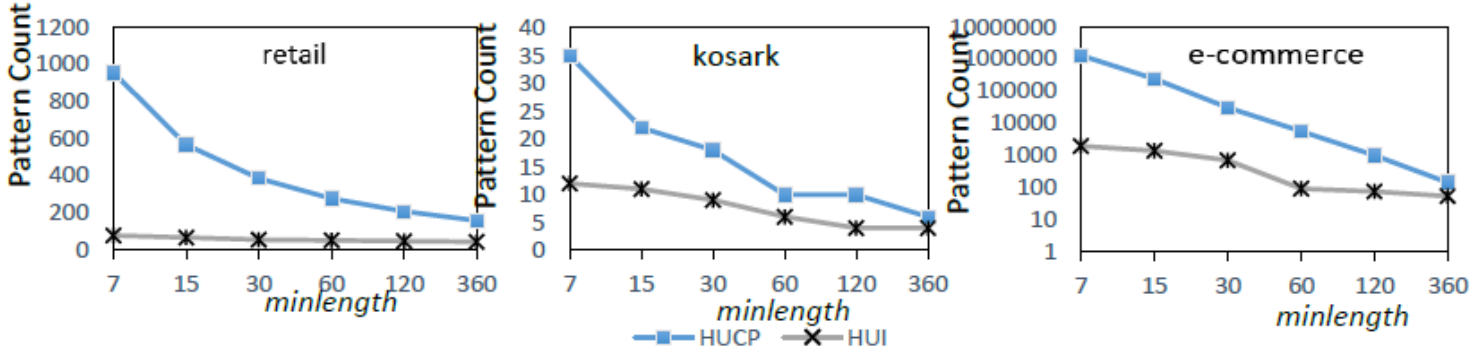
Dataset	Trans count	Item count	Average length	Type
$kosarak$	990,000	41,270	8.09	Long transaction
$retail$	88,162	16,470	10.3	sparse
$e\_commerce$	17,535	3,803	15.4	Real-life data

- We compared:
  - the **execution time** of HUCP-Miner with and without optimizations
  - the **number of patterns found** (**LHUIs** and **HUIs**)
- 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 HUCPs is much more than the number of HUIs in most cases.

Example: for $minMau = 16{,}000$ and $\text{winLength} = 90\ days$, the itemset $\{pink\ polkadot\ bag, black\ and\ white\ baroque\ bag\}$ has a positive trend on 2011/07/19 and a negative trend on 2011/11/02 where it generates a high utility.
And this itemset is not a HUI in the whole database for $minutil = minMau \times 373 = 5{,}968000$.

Conclusion

- New pattern type: **High Utility Change Points**
- New algorithm: **HUCP-Miner**
- **Results:**
  - optimizations can reduce runtime by half in some cases,
  - generally, there is much more HUCP than HUIs.
- **Future work:**
  - change points for other pattern mining problems,
  - discover concise representations of HUCPs.

Thank you. Questions?

SPMF

Open source Java data mining software, 130 algorithms
http://www.phillippe-fournier-viger.com/spmf/