Mining Correlated High-Utility Itemsets using the Bond Measure

Philippe Fournier-Viger[1], Jerry Chun-Wei Lin[2],
Quang-Huy Duong[3], Thu-Lan Dam[3,4],

[1] School of Natural Sciences and Humanities, Harbin Institute of Technology Shenzhen Graduate School, China
[2] School of Computer Science and Technology, Harbin Institute of Technology Shenzhen Graduate School, China
[3] Faculty of Information Technology, Ho Chi Minh city Industry and Trade College
[4] Faculty of Information Technology, University of Science, Vietnam
philippe.fournier-viger@umoncton.ca, jerrylin@ieee.org,
duytai@fit-hitu.edu.vn, lhbac@fit.hcmus.edu.vn

Topic of this talk

- The problem of discovering interesting and useful patterns in databases.
- In particular, extensions of the problem of ***high-utility itemset mining***

What is a transaction database?

- Let be a set of items **{a, b, c, d, e, …}** sold in a store.

- A ***transaction*** is a set of items bought by a customer.
- **Example:**

Transaction	items
T_1	{a, b, c, d, e}
T_2	{a, b, e}
T_3	{c, d, e}
T_4	{a, b, d, e}

four transactions

Discovering Frequent Patterns

- The **task of** ***frequent pattern mining*** was proposed by Agrawal (1993).
- **Input**: a transaction database and a parameter ***minsup*** ≥ 1.
- **Output**: the ***frequent itemsets*** (all sets of items appearing in at least ***minsup*** transactions).

An example

transaction database

Transaction	items
T_1	{a, b, c, d, e}
T_2	{a, b, e}
T_3	{c, d, e}
T_4	{a, b, d, e}

***minsup* = 2**

frequent itemsets

Itemset	Support
{e}	4
{d, e}	3
{b, d, e}	2
…	…

How to solve this problem?

The naïve approach:

- scan the database to count the frequency of **each** possible itemset.
  eg.: {a}, {a,b}, {a,c}, {a,d}, {a,e}, {a,b,c}, {a,b,d} … {b}, {b, c}, … {a,b,c,d,e}
- If $\boldsymbol{n}$ items, then $\mathbf{2^n - 1}$ possible itemsets.
- Thus, inefficient.

- **Several efficient algorithms:**
  - Apriori, FPGrowth, H-Mine, LCM, etc.

The “Apriori” property

Property (anti-monotonicity).

Let be itemsets **X** and **Y.** If **X** ⊂ **Y**, then the support of Y is less than or equal to the support of **X**.

Example

Transaction	items
T_1	{a, b, c, d, e}
T_2	{a, b, e}
T_3	{c, d, e}
T_4	{a, b, d, e}

The support of **{a,b}** is **3**.

Thus, supersets of **{a,b}** have a support ≤ **3.**

Limitations of frequent patterns

- **Frequent pattern mining** has **many** applications**.**
- However, it has important **limitations**
  - many frequent patterns are not interesting,
  - quantities of items in transactions must be 0 or 1
  - all items are considered as equally important (having the same weight)

High Utility Itemset Mining

- **A generalization of frequent pattern mining:**
  - items can appear more than once in a transaction (**e.g.** a customer may buy 3 bottles of milk)
  - items have a unit profit (**e.g.** a bottle of milk generates 1 $ of profit)
  - the goal is to find **patterns that generate a high profit**
- **Example**:
  - {caviar, wine} is a pattern that generates a high profit, although it is rare

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

Output

All high-utility itemsets (itemsets having a utility $\geq minutil$)
For example, if *minutil = 33$*, the high-utility itemsets are:

***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:

$$u(\{b,d,e\}) = \underbrace{(5x2)+(3x2)+(3x1)}_{\text{utility in transaction } T_1} + \underbrace{(4x2)+(2x3)+(1x3)}_{\text{utility in transaction } T_2} = \mathbf{36\$}$$

How to solve this problem?

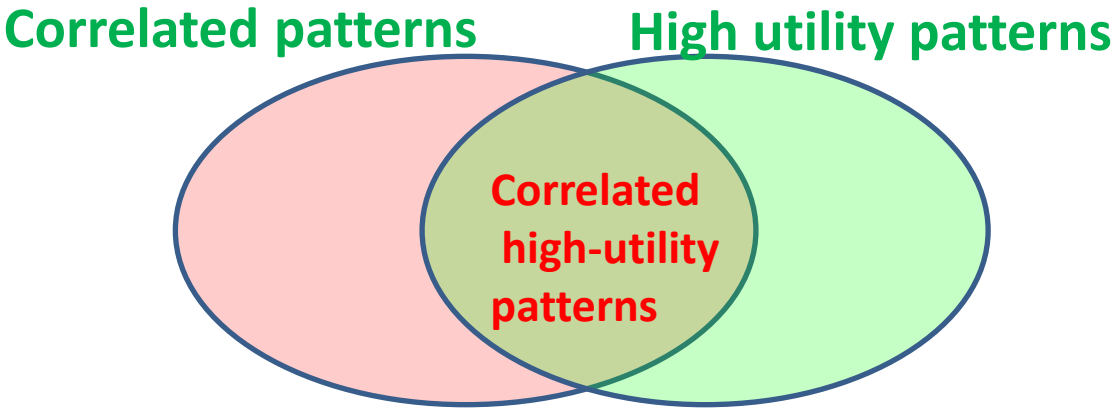
- **Several algorithms:**
  - **Two-Phase (PAKDD 2005),**
  - **IHUP (TKDE, 2010),**
  - **UP-Growth (KDD 2011),**
  - **HUI-Miner (CIKM 2012),**
  - **FHM (ISMIS 2014)**
  - **EFIM (2015)**
  - **mHUIMiner (2017)**
- **Key idea**: calculate an upper-bound on the utility of itemsets (**e.g.** the **TWU**) that respects the Apriori property to be able to prune the search space.

Problem

High-utility itemset mining

- is **useful** for discovering **profitable itemsets**.
- but **may discover many itemsets that are weakly correlated**
- **e.g.** bread with caviar has a high profit

We propose a new type of patterns:



How to detect if items are correlated?

Several approaches:

- using **statistical tests** to find productive itemsets (Webb et al., 2010)
- the ***affinity*** measure (Ahmed et al. 2011)
- the ***bond*** measure (Bouasker et al., 2015)

The *bond* of an itemset

- The **conjunctive support** of an itemset **X** in a database is the number of transactions that **contains X**.
- The **disjunctive support** of an itemset **X** in a database is the number of transactions that contains **any items from X**.
- The **bond** of an itemset **X** is defined as:

  bond(X) = conj_sup(X) / disj_sup(X).

Property 4 (Anti-monotonicity of the bond measure). Let X and Y be two itemsets such that $X \subseteq Y$. It follows that $bond(X) \geq bond(Y)$ [6].

Problem definition

- **Discovering all correlated high utility itemsets**, that is itemsets:
  - having a **utility** no less than a threshold ***minutil***
  - having a **bond** no less than a threshold ***minbond***

Example

Table 1: 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	$(b,2),(c,2),(e,1),(g,2)$

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

For example, if **minutil = 30** and **minbond = 0.5**, correlated high utility itemsets are :

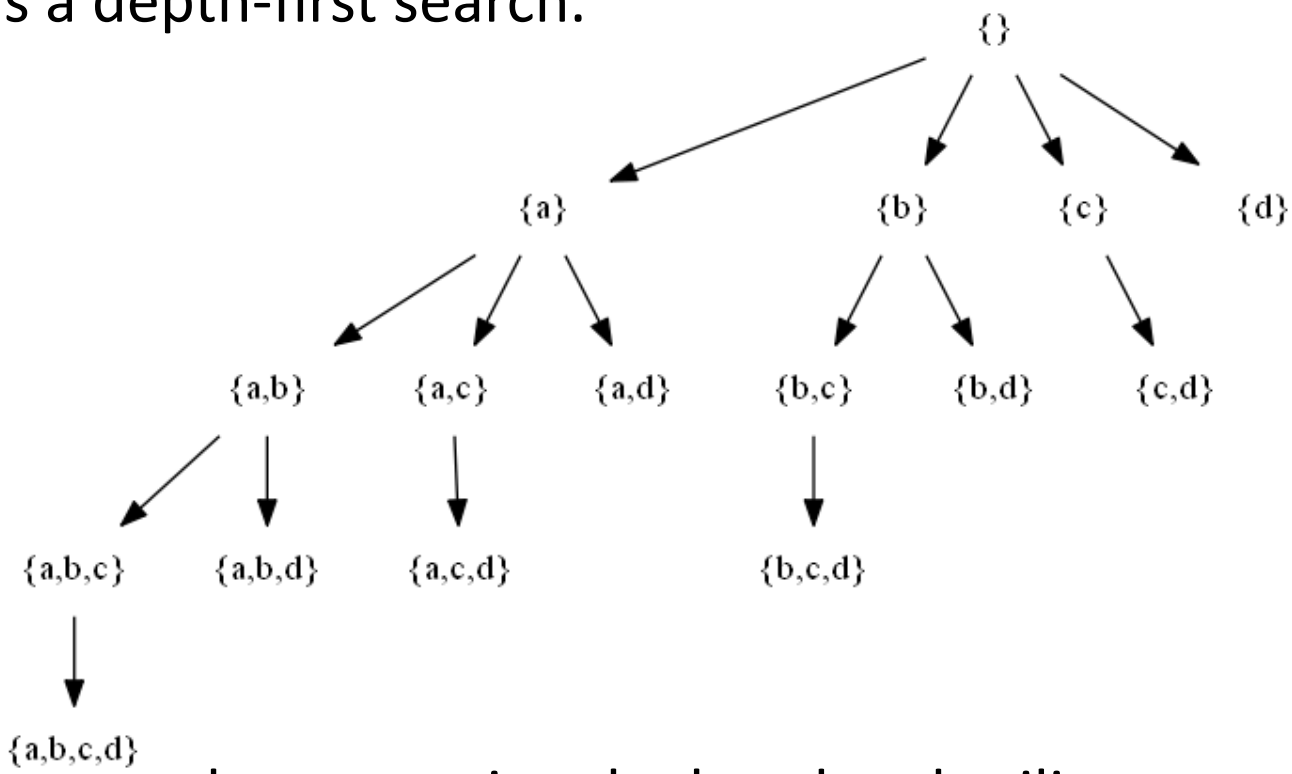
{b, d} bond = 2 / 4 = 0.5

{b,e} bond = 3 / 4 = 0.75

{b, c, e} bond = 3 / 5 = 0.6

The FCHM algorithm

- An algorithm for mining correlated high utility-itemsets
- It performs a depth-first search.



- It prune the search space using the bond and utility measures.
- **Key challenge**: how to calculate the bond of an itemset

The FCHM algorithm (cont'd)

- The algorithm is inspired by the **FHM** algorithm for high-utility itemset mining.
- **FCHM** annotates each itemset with a structure called utility-list.
- Moreover each itemset **X** is annotated with a **conjunctive bitvector** that stores the union of all items in X

e.g. the conj. bitvector of {a,c} is T_1, T_3, T_5, T_6 101011
the conj. bitvector of {d} is T_3, T_4, T_5 001110
the conj. bitvector of {a,c,d} is T_1, T_3, T_4, T_5, T_6 101111

The conjunctive bitvector of an itemset can be obtained by performing the union of the bitvectors of some of its subsets.

Additional optimizations

discover CHIs.

Strategy 1. Directly Outputting Single items (DOS). A first optimization is based on the observation that the bond of single items is always equal to 1. Thus, HUIs containing a single item can be directly output without calculating their bond.

Strategy 2. Pruning Supersets of Non correlated itemsets (PSN). Because the bond measure is anti-monotonic (Property 5), if the bond of an itemset Pxy is less than $minbond$, any extensions of Pxy should not be explored. Thus, Line 8 of Algorithm 1 is modified so that if $bond(Pxy) < minutil$ for an itemset Pxy, the utility-list of Pxy is not added to the set $ExtensionsOfPx$.

Strategy 3. Pruning using the Bond Matrix (PBM). The third strategy is introduced to avoid constructing the utility-list of an itemset Pxy, and prune all its extensions. During the second database scan, a novel structure named *Bond Matrix* is created to store the bond of all itemsets containing two items from I^*. The design of this structure is similar to the EUCS. The Bond Matrix is formally defined as follows. The Bond Matrix is a set of triples of the form $(a, b, c) \in I^* \times I^* \times \mathbb{R}$. A triple (a,b,c) indicates that $bond(\{a, b\}) = c$. Note that it only stores tuples of the form (a, b, c) such that $c \neq 0$. For example, the Bond Matrix for the running example is shown in Fig. 2.

Then, Line 5 of Algorithm 1 is modified to add the condition that the utility-list of Pxy should only be built if $\exists(x, y, c) \in BondMatrix$ such that $c \geq minbond$. It can be easily seen that this pruning condition preserve the correctness and completeness of FCHM, because any superset of an itemset $\{x, y\}$ such that $bond(\{x, y\}) < minbond$ is not a CHIs by Property 5.

Additional optimizations

Strategy 4. Abandoning Utility-List construction early (AUL). The fourth strategy introduced in FCHM is to stop constructing the utility-list of an itemset if some specific condition is met, indicating that the itemset may not be a CHI. The strategy is based on the following novel observation:

Property 6 (Required conjunctive support for an extension Pxy). An itemset Pxy is a correlated itemset only if its conjunctive support is no less than the value $lowerBound(Pxy) = \lceil |consup(Pxy)| * minBond) \rceil$ transactions.

This property is directly derived from the definition of the bond measure, and proof is therefore omitted. Now, consider the construction of the utility-list of an itemset Pxy by the Construct procedure (Algorithm 3). As mentioned, a first modification to this procedure is to create the disjunctive bit vector of Pxy by performing the OR operation with the bit vectors of Px and Py before Line 1. This allows obtaining the value $consup(Pxy) = |bv(Pxy)|$. A second modification is to create a variable $maxSupport$ that is initialized to the conjunctive support of Px ($consup(Px) = |bv(Px)|$, before Line 1. The third modification is to the following lines, where the utility-list of Pxy is constructed by checking if each tuple in the utility-lists of Px appears in the utility-list of Py (Line 3). For each tuple not appearing in Py, the variable $maxSupport$ is decremented by 1. If $maxSupport$ is smaller than $lowerBound(Pxy)$, the construction of the utility-list of Pxy can be stopped because the conjunctive support of Pxy will not be higher than $lowerBound(Pxy)$. Thus Pxy is not a CHI by Property 6, and its extensions can also be ignored by Property 5. As it will be shown in the experiment, this strategy is very effective, and decrease execution time and memory usage by stopping utility-list construction early. A similar pruning strategy based on the utility of itemset Pxy instead of the bond measure is also integrated in FCHM. This strategy is called the LA-prune strategy and was introduced in HUP-Miner [8], and it is thus not described here.

Experimental Evaluation

Datasets' characterictics

Dataset	transaction count	distinct item count	average transaction length
Retail	88,162	16,470	10.30
Kosarak	990,000	41,270	8.09
Foodmart	1,559	4,141	4.4
Mushroom	8,124	120	23

Retail, **Foodmart** are real-life transaction datasets from retail stores.

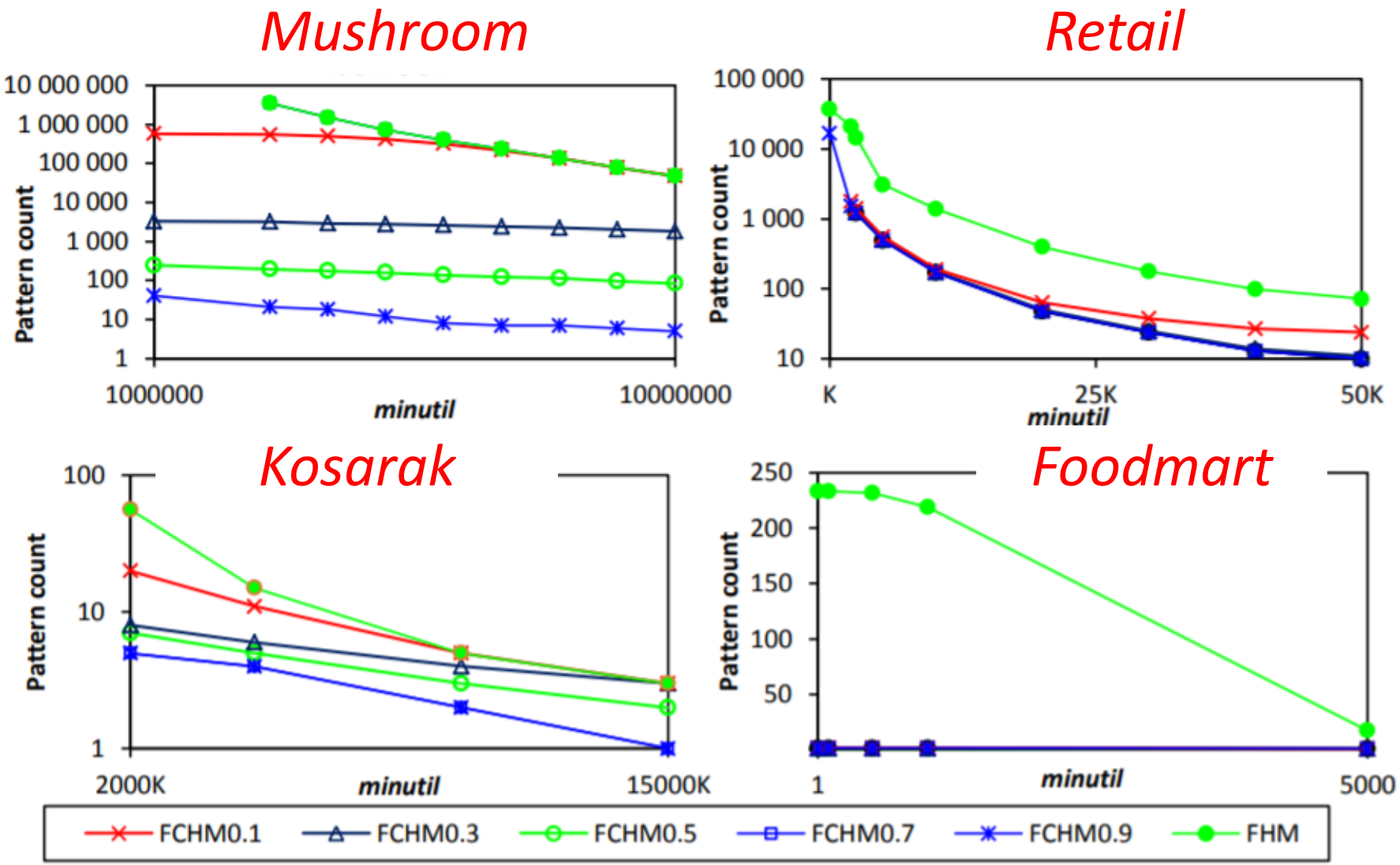
Mushroom is a dense dataset with long transactions
Kosarak is a large sparse dataset

Experimental Evaluation

- **FCHM** was run with five different **minbond** threshold values (0.1, 0.3, 0.5, 0.7 and 0.9).
- It was compared with the **FHM** algorithm for high utility itemset mining

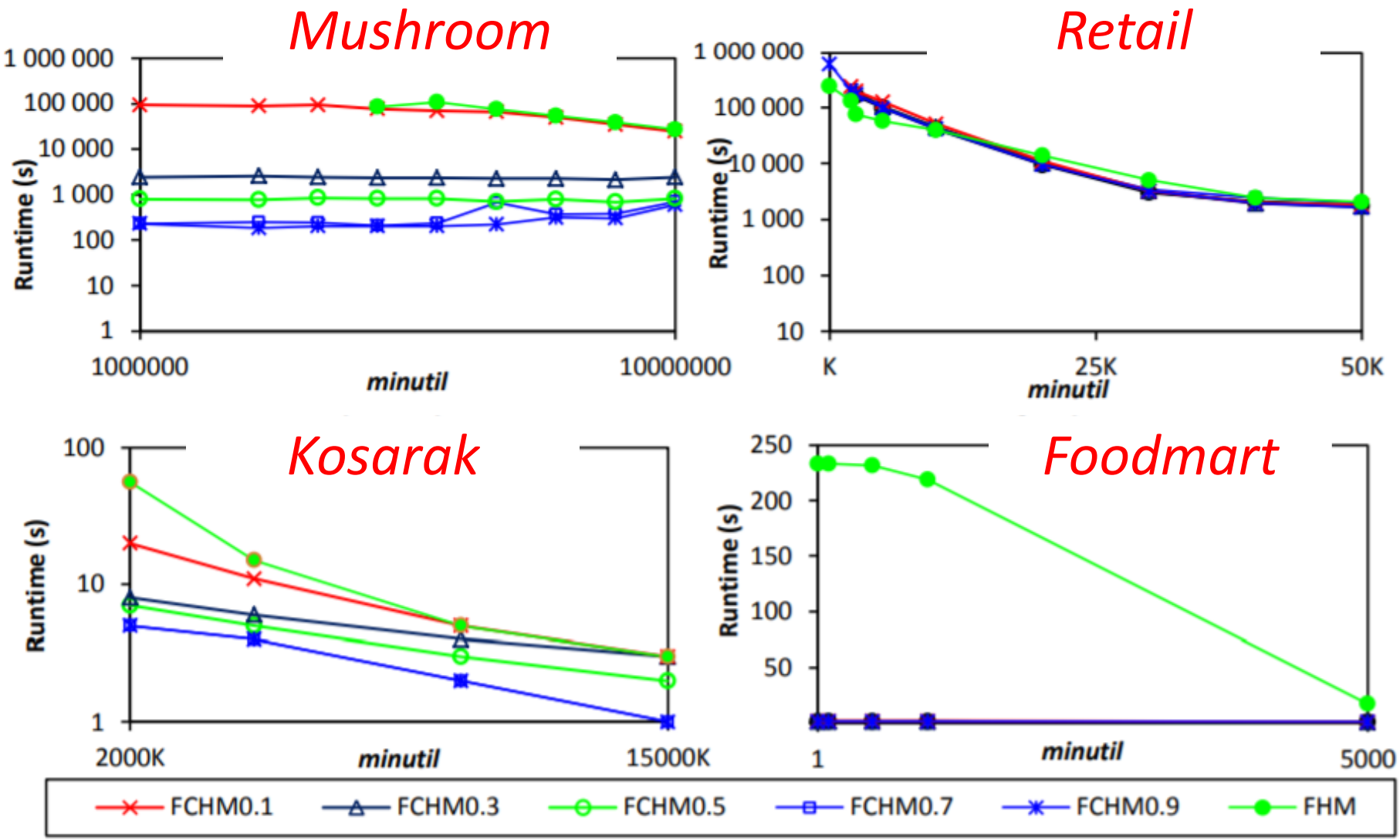
12 GB of RAM, Java, Windows 7, 64 bit Core i5 Processor

Number of Pattern found



The FCHM algorithm can filter many non correlated patterns.

Execution times



The FCHM algorithm can be up to 100 times faster than FHM

Other results

Memory consumption was also compared, although detailed results are not shown as a figure due to space limitation. It was observed that FCHM can use up to 30 times less memory than FHM depending on how the *minbond* threshold is set. For example, on *mushroom* and $minutil = 5,000,000$, FHM, FCHM0.9, FCHM0.7, FCHM0.5, FCHM0.3, and FCHM0.1, respectively consume 666 MB, 424 MB, 136 MB, 59 MB, 26 MB and 20 MB.

Lastly, the efficiency of the proposed PBM and AUL strategies was also assessed. Detailed results are not shown due to space limitation. But these strategies were shown to prune a huge amount of candidates for most datasets, and it greatly increases when *minbond* is increased. For example, on *mushroom* and $minutil = 5,000,000$, FHM, FCHM0.9, FCHM0.7, FCHM0.5, FCHM0.3, and FCHM0.1, visit 1,282,377, 536,136, 5,254, 1,160, 883, and 848 candidates.

Conclusion

- Contributions:
  - New type of pattern: **correlated high-utility itemsets**
  - A novel algorithm, named **FCHM** with novel pruning strategies
- Experimental results:
  - **FCHM** eliminate a large number of non correlated patterns.
  - Can be much faster than FHM in many cases.
- Source code and datasets available as part of the **SPMF data mining library (**GPL 3).

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

Thank you. Questions?

Open source Java data mining software, 133 algorithms
http://www.phillippe-fournier-viger.com/spmf/

SPMF

An Open-Source Data Mining Library

Introduction

Algorithms

Download

Documentation

Datasets

FAQ

License

Contributors

Citations

Performance

Developers' guide

Forum

Mailing-list

Blog

Introduction

SPMF is an **open-source data mining mining library** written in **Java**, specialized in **pattern mining**.

It is distributed under the **GPL v3 license**.

It offers implementations of **120 data mining algorithms** for:

- **association rule mining,**
- **itemset mining,**
- **sequential pattern mining,**
- **sequential rule mining,**
- **sequence prediction**,
- **periodic pattern mining,**
- **high-utility pattern mining,**
- **clustering** and **classification**

The **source code** of each algorithm can be easily integrated in other Java software.

Moreover, SPMF can be used as a **standalone program** with a simple user interface or from the **command line**.

SPMF is fast and lightweight (no dependencies to other libraries).

The current version is **v0.99j** and was released the **16th June 2016.**

http://www.philippe-fournier-viger.com/spmf/

Running an algorithm

SPMF v0.99j

Choose an algorithm: CM-SPAM ?

Choose input file: snake_192_converted.txt ...

Set output file: test.txt ...

Choose minsup (%): 0.96 (e.g. 0.5 or 50%)

Min pattern length (optional): 4 (e.g. 1 items)

Max pattern length (optional): (e.g. 10 items)

Required items (optional): (e.g. 1,2,3)

Max gap (optional): (e.g. 1 item)

Show sequence ids? (optional): (default: false)

Open output file:

☑ using SPMF viewer ☐ using text editor

Run algorithm

```
Algorithm is running...
============  CM-SPAM v0.97 - STATISTICS =============
 Total time ~ 135 ms
 Frequent sequences count : 447
 Max memory (mb) : 39.53382110595703447
minsup 157
Intersection count 2141
=================================================
```

Discovered patterns

SPMF - Pattern vizualization tool

Patterns:

Pattern	#SUP:
2 -1 2 -1 2 -1 2 -1	163
2 -1 2 -1 2 -1 2 -1 2 -1	160
[illegible] -1 2 -1 2 -1 2 -1 2 -1 2 -1	157
[illegible] -1 2 -1 2 -1 10 -1	162
[illegible] 1 2 -1 3 -1	160
[illegible] 2 -1 2 -1 6 -1	163
2 -1 2 -1 2 -1 6 -1 2 -1	163
2 -1 2 -1 2 -1 10 -1	163
2 -1 2 -1 2 -1 10 -1 2 -1	160
2 -1 2 -1 2 -1 10 -1 2 -1 2 -1	158
2 -1 2 -1 2 -1 10 -1 3 -1	157
2 -1 2 -1 2 -1 10 -1 6 -1	160
2 -1 2 -1 2 -1 10 -1 17 -1	161
2 -1 2 -1 2 -1 10 -1 17 -1 6 -1	158
2 -1 2 -1 2 -1 10 -1 19 -1	158
2 -1 2 -1 2 -1 15 -1	161
2 -1 2 -1 2 -1 15 -1 2 -1	160
2 -1 2 -1 2 -1 17 -1	163
2 -1 2 -1 2 -1 17 -1 2 -1	159
2 -1 2 -1 2 -1 17 -1 6 -1	161
2 -1 2 -1 2 -1 17 -1 6 -1 2 -1	158
2 -1 2 -1 2 -1 19 -1	159
2 -1 2 -1 6 -1 2 -1	163
2 -1 2 -1 6 -1 2 -1 2 -1	158
2 -1 2 -1 6 -1 2 -1 6 -1	163
2 -1 2 -1 6 -1 2 -1 10 -1	158
2 -1 2 -1 6 -1 6 -1	163

Search:

Apply filter(s):

Add a filter

Remove selected filter

Remove all filters

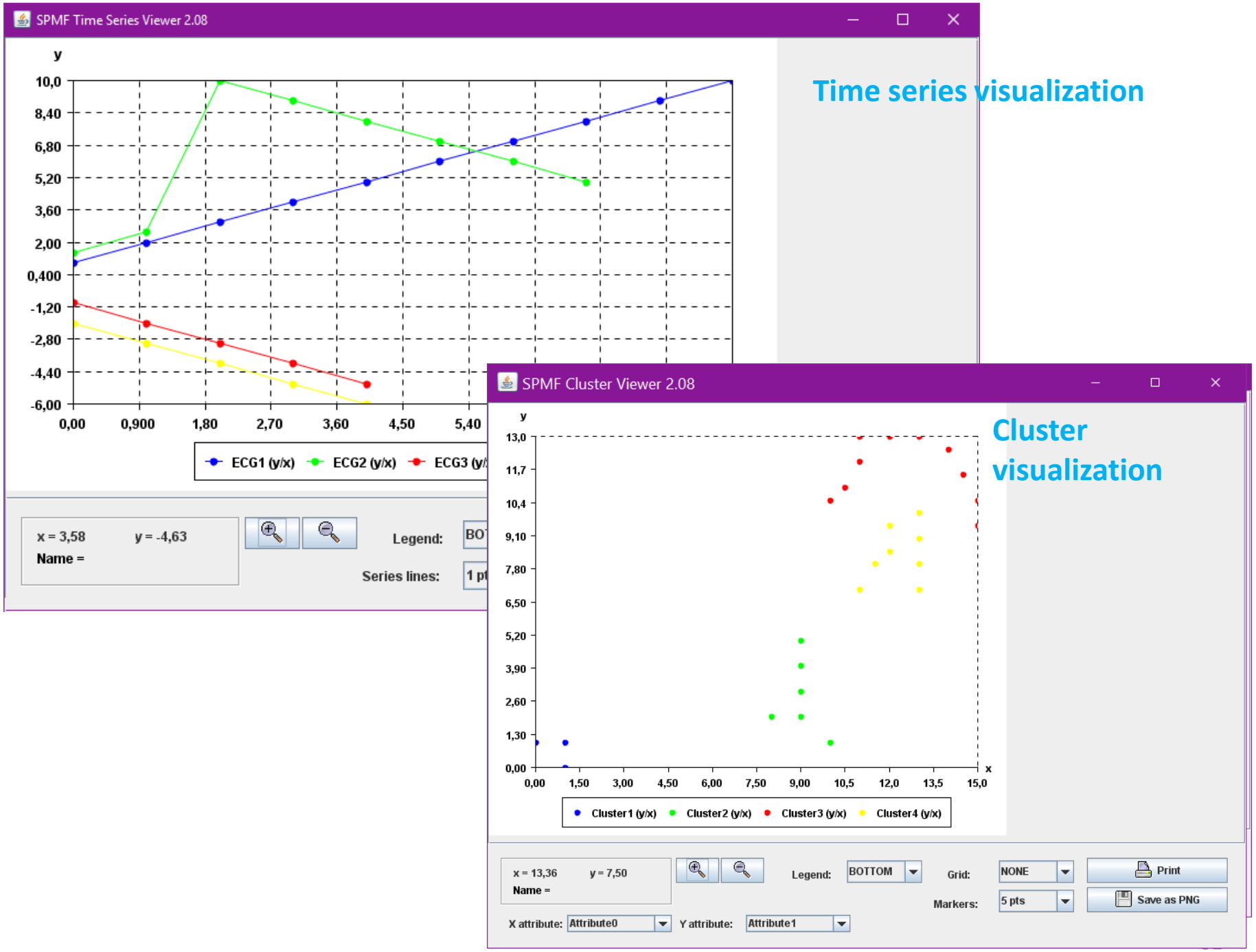
Export current view to:

SPMF format

Number of patterns: 447

File name: test.txt File size (MB): 0,0152 Last modified: 2016-08-05, 11:08

http://www.philippe-fournier-viger.com/spmf/



http://www.philippe-fournier-viger.com/spmf/

Springer

CALL FOR CHAPTERS

High-Utility Pattern Mining: Theory, Algorithms and Applications

Editors: Philippe Fournier-Viger, Chun-Wei Lin, Roger Nkambou, Bay Vo

An edited book to be published by **Springer** in **2018**

Chapter proposal deadline:	1st October 2017
Proposal acceptance date:	10th October 2017
Full chapter submission deadline:	15th January 2017
Planned publication date:	1st July 2018

Introduction

High utility pattern mining is a popular and emerging research area in the field of data mining, which consists of discovering patterns of high importance in databases. The importance of patterns can be expressed in terms of various criteria such as the utility, weight or importance of patterns. One of the major applications of utility mining is the analysis of customer transaction databases to discover sets of items that generate a high profit when purchased together. This book provides an introduction to this field, reviews state-of-the-art techniques, and discusses recent advances.

Book aims

- Presents an overview of the theory and core methods used in utility mining

References

- Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Proceedings of the International Conference on Very Large Databases, pp. 487–499 (1994)
- Fournier-Viger, P., Wu, C.-W., Zida, S., Tseng, V.S.: **FHM: faster high-utility itemset mining using estimated utility co-occurrence pruning.** In: Andreasen, T., Christiansen, H., Cubero, J.-C., Ra´s, Z.W. (eds.) ISMIS 2014. LNCS, vol. 8502, pp. 83–92. Springer, Heidelberg (2014)
- Fournier-Viger, P., Gomariz, A., Gueniche, T., Soltani, A., Wu, C., Tseng, V.S.: SPMF: a Java open-source pattern mining library. J. Mach. Learn. Res. (JMLR) 15, 3389–3393 (2014)
- Pei, J., Han, J.: Constrained frequent pattern mining: a pattern-growth view. ACM SIGKDD Explor. Newsl. 4(1), 31–39 (2012)
- Lan, G.C., Hong, T.P., Tseng, V.S.: An efficient projection-based indexing approach for mining high utility itemsets. Knowl. Inf. Syst. 38(1), 85–107 (2014)
- Krishnamoorthy, S.: Pruning strategies for mining high utility itemsets. Expert Syst. Appl. 42(5), 2371–2381 (2015)

References (cont'd)

- Lin, J.C.-W., Gan, W., Hong, T.-P., Pan, J.-S.: Incrementally updating high-utility itemsets with transaction insertion. In: Luo, X., Yu, J.X., Li, Z. (eds.) ADMA 2014. LNCS, vol. 8933, pp. 44–56. Springer, Heidelberg (2014)
- Song, W., Liu, Y., Li, J.: BAHUI: fast and memory efficient mining of high utility itemsets based on bitmap. Int. J. Data Warehous. Min. 10(1), 1–15 (2014)
- Liu, M., Qu, J.: Mining high utility itemsets without candidate generation. In: Proceedings of the 22nd ACM International Conference on Information and Knowledge Management, pp. 55–64 (2012)
- Liu, Y., Liao, W., Choudhary, A.K.: A two-phase algorithm for fast discovery of high utility itemsets. In: Ho, T.-B., Cheung, D., Liu, H. (eds.) PAKDD 2005. LNCS (LNAI), vol. 3518, pp. 689–695. Springer, Heidelberg (2005)
- Tseng, V.S., Shie, B.-E., Wu, C.-W., Yu, P.S.: Efficient algorithms for mining high utility itemsets from transactional databases. IEEE Trans. Knowl. Data Eng. 25(8), 1772–1786 (2013)
- Zida, S., Fournier-Viger, P., Wu, C.-W., Lin, J.C.W., Tseng, V.S.: Efficient mining of high utility sequential rules. In: Proceedings of the 11th International Conference on Machine Learning and Data Mining, pp. 1–15 (2015)
- Zida, S., Fournier-Viger, P., Lin, J.C.-W., Wu, C.-W., Tseng, V.S.: EFIM: a highly efficient algorithm for high-utility itemset mining. In: Sidorov, G., Galicia-Haro, S.N. (eds.) MICAI 2015. LNCS, vol. 9413, pp. 530–546. Springer, Heidelberg (2015). doi:10.1007/978-3-319-27060-9 44
- Fournier-Viger, P., Lin, C. W., Dinh, T., Le, H. B. (2016). **Mining Correlated High-Utility Itemsets Using the Bond Measure**. Proc. 11 th International Conference on Hybrid Artificial Intelligence Systems (HAIS 2016), Springer LNAI, pp.53-65.
- Fournier-Viger, P., Lin, C.W., Duong, Q.-H., Dam, T.-L. (2016). **PHM: Mining Periodic High-Utility Itemsets**. Proc. 16th Industrial Conference on Data Mining. Springer LNAI 9728, pp. 64-79
- Fournier-Viger, P., Zida, S. (2015). **FOSHU: Faster On-Shelf High Utility Itemset Mining– with or without negative unit profit**. Proc. 30th Symposium on Applied Computing (ACM SAC 2015). ACM Press, pp. 857-864.