PHM: Mining Periodic High-utility Itemsets

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] College of Computer Science and Electronic Engineering, Hunan University, China
[4] Faculty of Information Technology, Hanoi University of Industry, Vietnam

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 $\text{utility} \geq minutil$)

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:

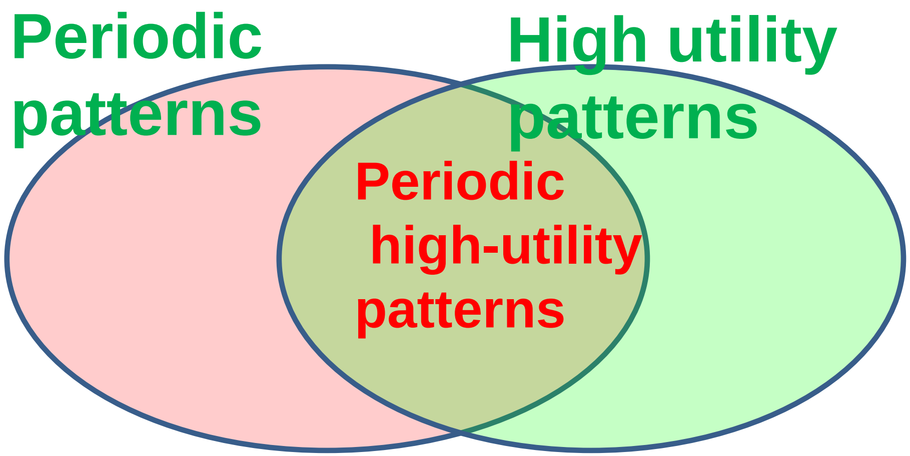
$$u(\{b,d,e\}) = \underbrace{(5\mathrm{x}2)+(3\mathrm{x}2)+(3\mathrm{x}1)}_{\text{utility in transaction } T_1} + \underbrace{(4\mathrm{x}2)+(2\mathrm{x}3)+(1\mathrm{x}3)}_{\text{utility in transaction } T_2} = \mathbf{36\$}$$

Problem

High-utility itemset mining

- is **useful** for discovering **profitable itemsets**.
- but **not designed for** discovering **recurring customer behavior**
- **e.g.** a customer buy {wine, cheese} every week

We propose a new type of patterns:



How to measure the periodicity?

Several studies:

- PFP-Tree, MKTPP, ITL-Tree, PF-tree, MaxCPF
- In general, a **periodic pattern** has no period greater than a maximum periodicity threshold (*maxPer*), set by the user.

Period of an itemset

The number of transactions between each occurrence of an itemset

TID	Transaction
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	$(b, 2), (c, 2), (e, 1)$

e.g. The periods of the itemset **{a,c}** are: **1,2,2,1,1**

The maximum period of **{a,c}**: **2**

Period of an itemset

The number of transactions between each occurrence of an itemset

TID	Transaction
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	$\{b, c, e\}$

e.g. The periods of the itemset **{a,c}** are: **1,2,2,1,1**

The maximum period of **{a,c}**: **2**

Limitation

- An itemset is automatically discarded as non periodic if it has a **single period** of length greater than the *maxPer* threshold

- **Our solution**: two novel measures:
  - Average periodicity
  - Minimum periodicity (which excludes the first and last periods)

Novel definition of periodic pattern

An itemset X is **periodic** if:

- *minAvg ≤ avgper(X) ≤ maxAvg*
- *Minper(X) ≥ minPer*
- *Maxper(X) ≤ maxper*

where minAvg, maxAvg, minPer, maxPer are parameters set by the user.

These new parameters give more flexibility to the user.

Example

	Number of occurrences	Minimum periodicity	Maximum periodicity	Average periodicity
Itemset	support s(X)	minper(X)	maxper(X)	avgper(X)
$\{b\}$	3	1	3	1.75
$\{b, e\}$	3	1	3	1.75
$\{b, c, e\}$	3	1	3	1.75
$\{b, c\}$	3	1	3	1.75
$\{d\}$	3	1	3	1.75
$\{c, d\}$	3	1	3	1.75
$\{a\}$	4	1	2	1.4
$\{a, c\}$	4	1	2	1.4
$\{e\}$	5	1	2	1.17
$\{c, e\}$	4	1	3	1.4
$\{c\}$	6	1	2	1.0

Theoretical results

Lemma 1 (Relationship between average periodicity and support). *Let X be an itemset appearing in a database D. An alternative and equivalent way of calculating the average periodicity of X is $avgper(X) = |D|/(|g(X)|+1)$.*

Lemma 2 (Monotonicity of the average periodicity). Let X and Y be itemsets such that $X \subset Y$. It follows that $avgper(Y) \geq avgper(X)$.

Lemma 3 (Monotonicity of the minimum periodicity). Let X and Y be itemsets such that $X \subset Y$. It follows that $minper(Y) \geq minper(X)$.

Lemma 4 (Monotonicity of the maximum periodicity). Let X and Y be itemsets such that $X \subset Y$. It follows that $maxper(Y) \geq maxper(X)$ [12].

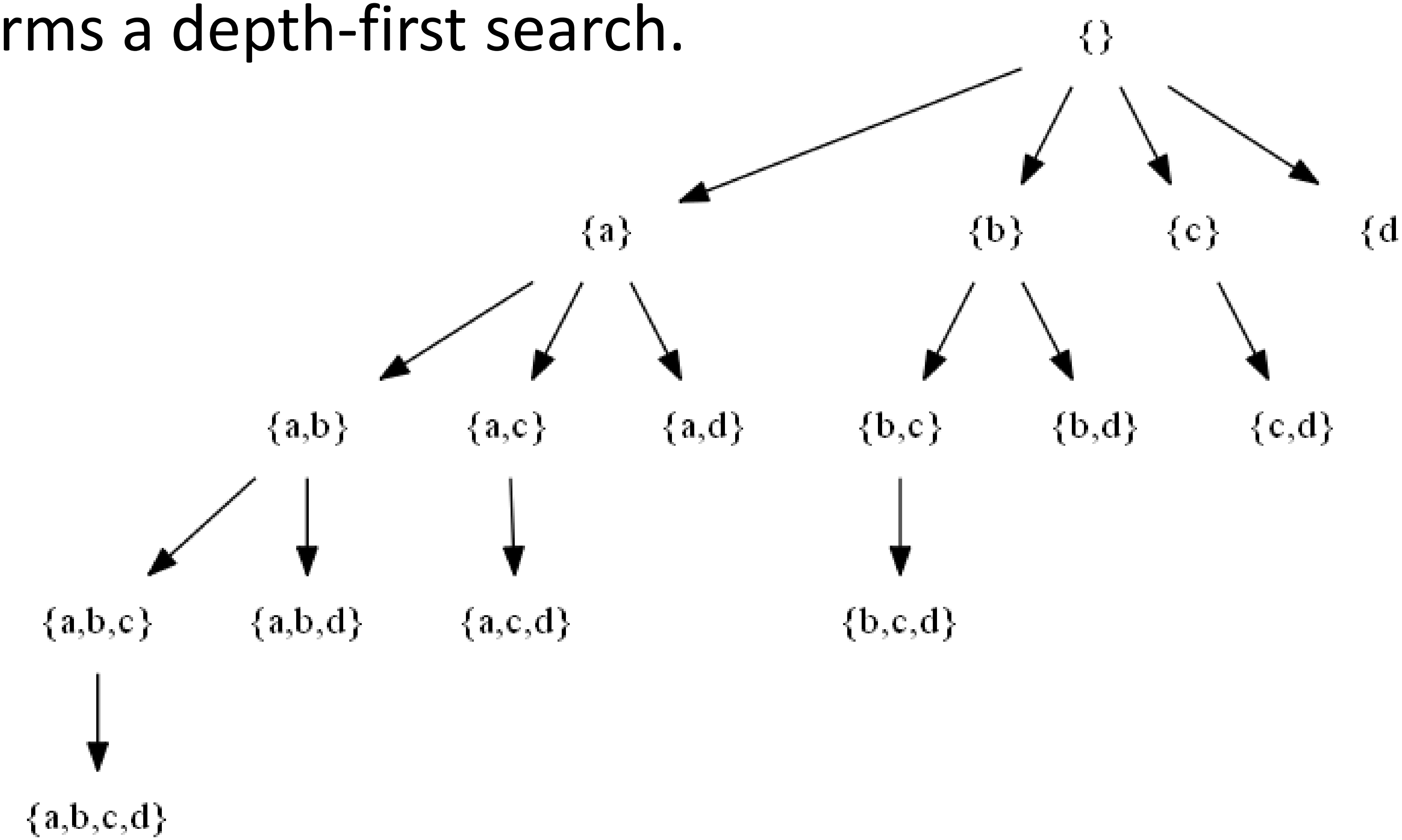
Theorem 3 (Maximum periodicity pruning). Let X be an itemset appearing in a database D. X and its supersets are not PHUIs if $maxper(X) > maxPer$.

Theorem 4 (Average periodicity pruning). Let X be an itemset appearing in a database D. X is not a PHUI as well as all of its supersets if $avgper(X) > maxAvg$, or equivalently if $|g(X)| < (|D|/maxAvg) - 1$.

First lemma is an efficient way of calculating the average periodicity.
Theorem 3 & 4 are used to reduce the search space.

The PHM algorithm

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



- It applies the theorems to prune the search space.

The PHM algorithm (cont'd)

- The algorithm is inpsired by the **FHM** algorithm for high-utility itemset mining.
- **PHM** annotates each itemset with its list of transactions (tid-list)

  e.g. the tid-list of {a,c} is $T_{1,}T_{3,}\ T_{5,}\ T_6$
  the tid-list of {d} is $T_{3,}\ T_{4,}\ T_5$
  the tid-list of {a,c,d} is $T_{3,}\ T_5$
- Tid-lists allows quickly calculating the periods of any itemset.
- The tid-list of any itemset can be calculated by intersecting tid-lists of smaller itemsets.

Pseudocode

Algorithm 1: The PHM algorithm

input : D: a transaction database, $minutil$, $minAvg$, $maxAvg$, $minPer$ and $maxPer$: the thresholds
output: the set of periodic high-utility itemsets

Scan D once to calculate $TWU(\{i\})$, $minper(\{i\})$, $maxper(\{i\})$, and $|g(\{i\})|$ for each item $i \in I$;
$\gamma \leftarrow (|D|/maxAvg) - 1$;
$I^* \leftarrow$ each item i such that TWU$(i) \geq minutil$, $|g(\{i\})| \geq \gamma$ and $maxper(\{i\}) \leq maxPer$;
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 $EUCS$ structure;
`Search` ($\emptyset$, I^*, γ, $minutil$, $minAvg$, $minPer$, $maxPer$, $EUCS$, $|D|$);

Algorithm 2: The *Search* procedure

input : P: an itemset, $ExtensionsOfP$: a set of extensions of P, γ, $minut$ $minAvg$, $minPer$, $maxPer$, the $EUCS$ structure, $|D|$
output: the set of periodic high-utility itemsets

foreach *itemset* $Px \in ExtensionsOfP$ **do**
  $avgperPx \leftarrow |D|/(|Px.utilitylist| + 1)$;
  if $SUM(Pxy.utilitylist.iutils) \geq minutil \wedge minAvg \leq avgperPx \leq maxAvg \wedge Px.utilitylist.minp \geq minPer \wedge Px.utilitylist.maxp \leq maxPer \wedge$ **then** output Px;
  if $SUM(Px.utilitylist.iutils) + SUM(Px.utilitylist.rutils) \geq minutil \wedge avgperPx \geq \gamma$ *and* $Px.utilitylist.maxp \leq maxPer$ **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);
        $ExtensionsOfPx \leftarrow ExtensionsOfPx \cup \{Pxy\}$;
      end
    end
    `Search` (Px, $ExtensionsOfPx$, γ, $minutil$, $minAvg$, $minPer$, max $EUCS$, $|D|$);
  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
    $period_{exy} \leftarrow calculatePeriod($exy.tid,UtilityListOfPxy$)$;
    $UpdateMinPerMaxPer(UtilityListOfPxy, period_{exy})$;
    $UtilityListOfPxy \leftarrow UtilityListOfPxy \cup \{exy\}$;
  end
end
return $UtilityListPxy$;

Experimental Evaluation

Datasets' characterictics

Dataset	transaction count	distinct item count	average transaction length
Retail	88,162	16,470	10.30
Chainstore	1,112,949	46,086	7.26
Foodmart	1,559	4,141	4.4
Mushroom	8,124	120	23

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

Mushroom is a dense dataset with long transactions

Experimental Evaluation

Compared algorithms

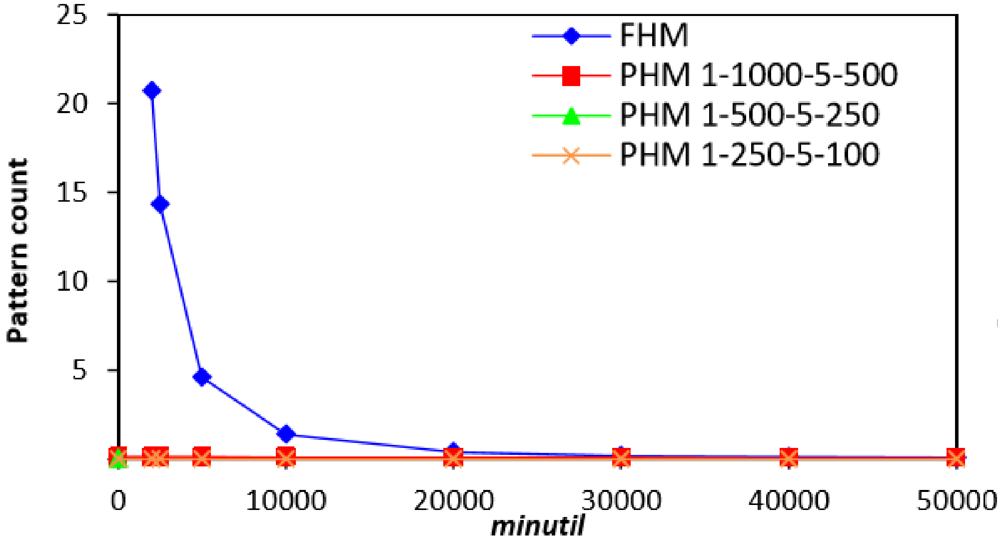
- We compared PHM with the state-of-the-art FHM algorithm for high-utility itemset mining.

- FHM find all high-utility itemsets
- PHM V-W-X-Y denotes the PHM algorithm with minper = V, maxper = W, minAvg = X, and maxAV G = Y .

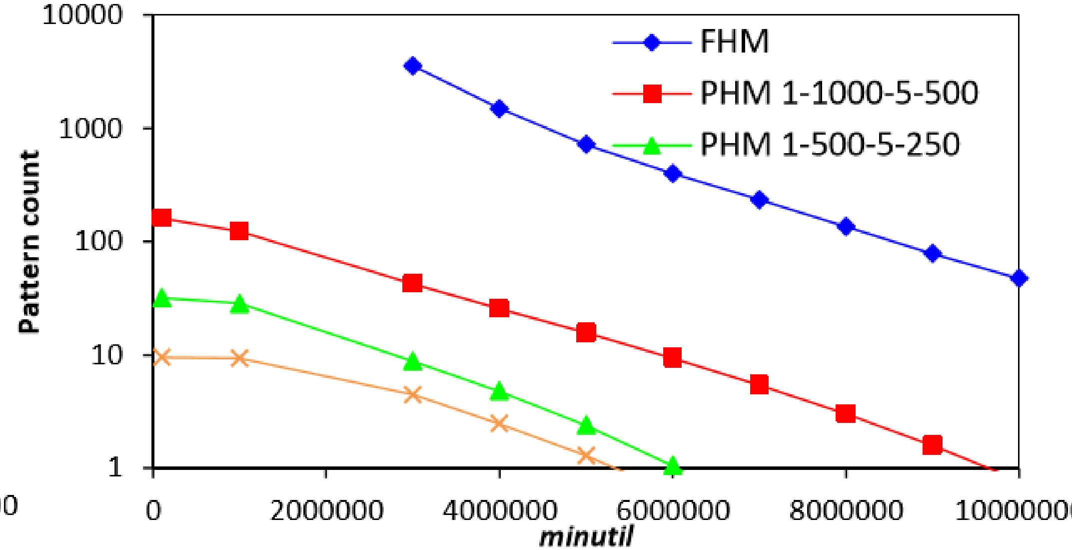
12 GB of RAM, Java, Windows 7, 64 bit Core i5 Processor

Number of Pattern found

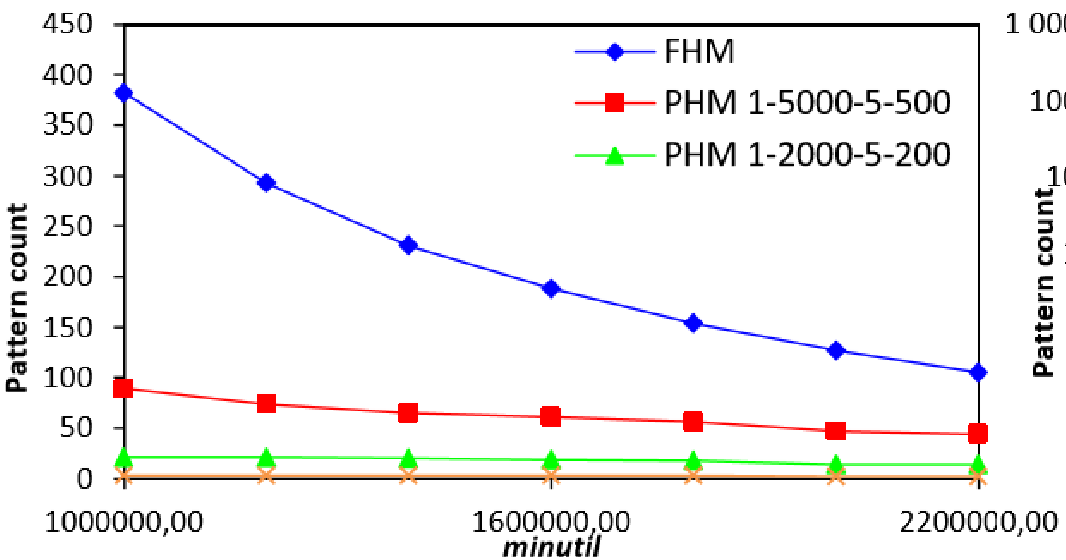
Retail



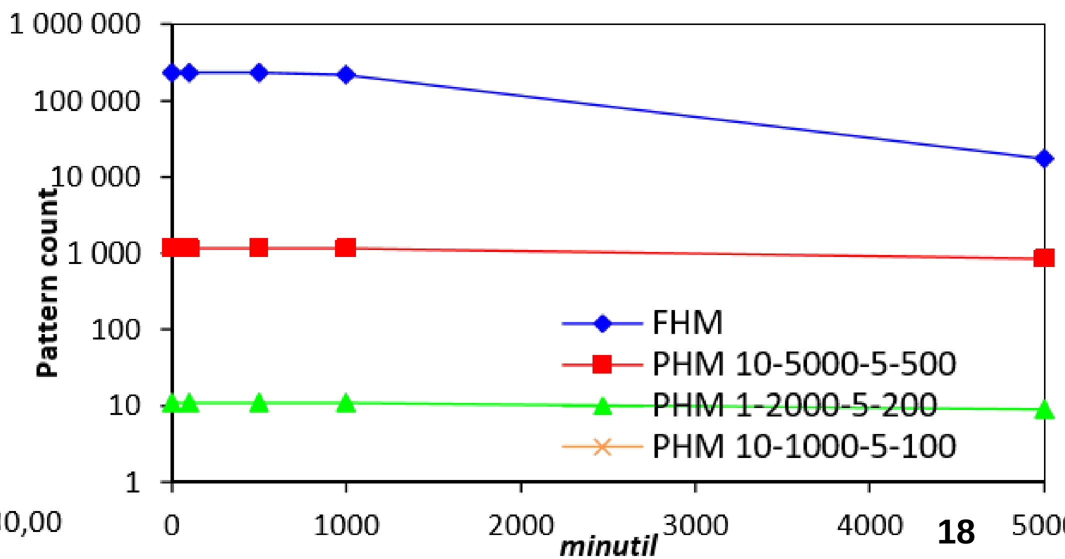
Mushroom



Chainstore



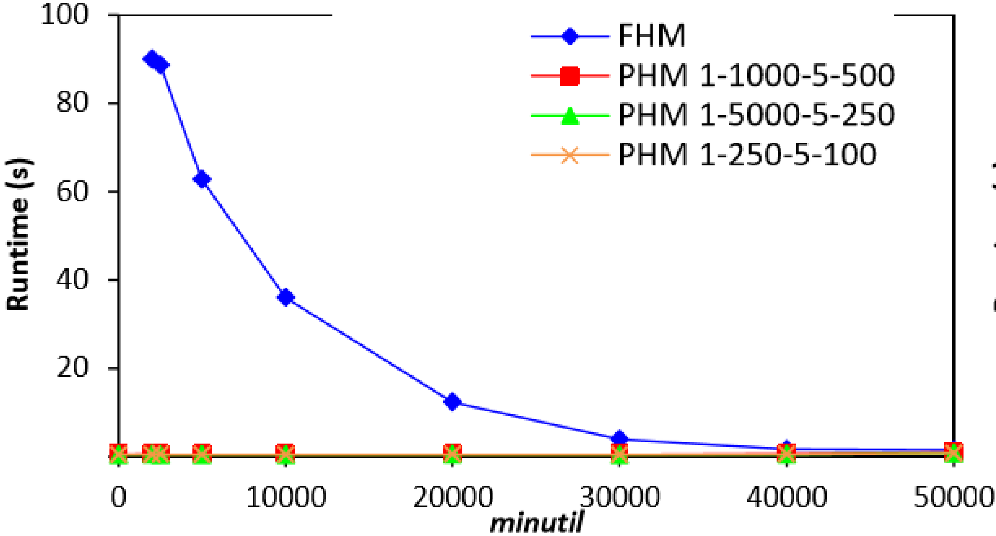
Foodmart



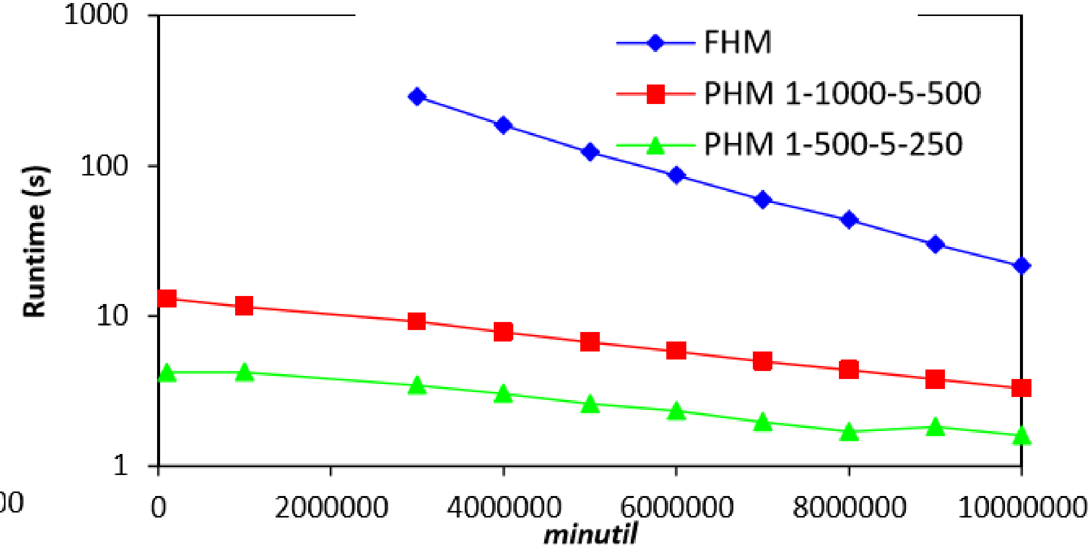
The PHM algorithm can filter many non periodic patterns.

Execution times

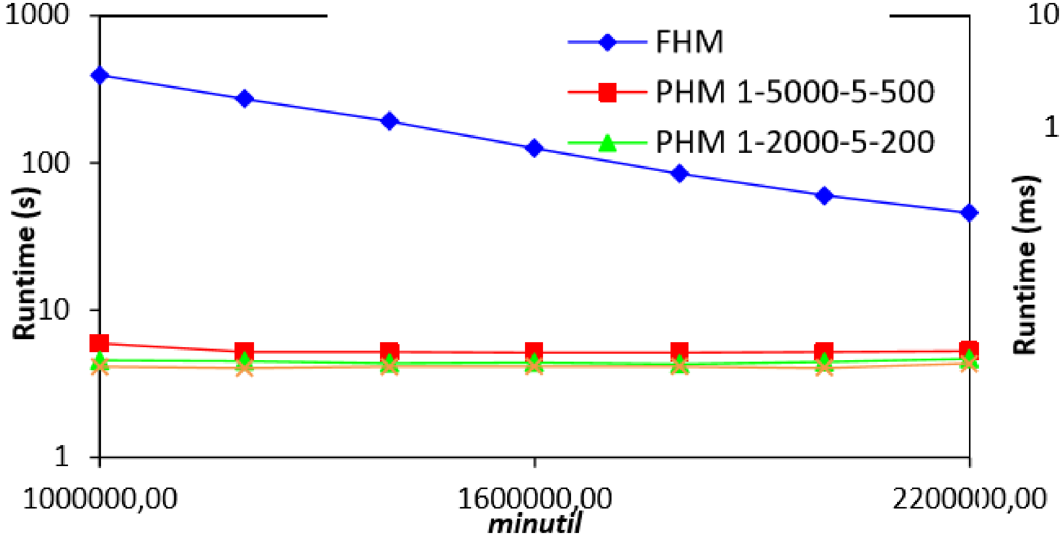
Retail



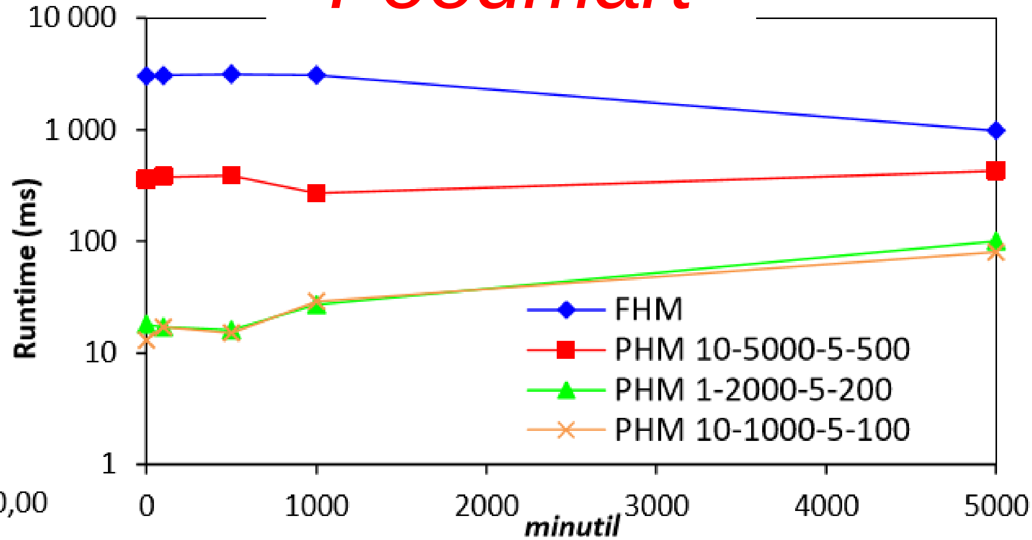
Mushroom



Chainstore



Foodmart



PHM can be much faster than FHM because it filters many non periodic patterns

Other observations

- Some **interesting patterns were found**. For example: products 32,48, and 39 are periodically bought with an average periodicity of 16.32, a minimum periodicity of 1 and a maximum periodicity of 170.
- PHM can use up to 10 less **memory** then **FHM**.
  - For example, on Chainstore and minutil = 1,000,000, FHM and PHM 1-5000-5-500 respectively consumes 1,631 MB and 159 MB of memory.

Conclusion

- Contributions:
  - New type of pattern: **periodic high-utility itemsets**
  - Two new periodicity measures: **average periodicity** and **minimum periodicity,** and their properties.
  - A novel algorithm, named **PHM**
- Experimental results:
  - **PHM** eliminate a large number of non-periodic 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, **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/

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.**