PFPM: Discovering Periodic Frequent Patterns with Novel Periodicity Measures

Philippe Fournier-Viger[a,*], Chun-Wei Lin[b], Quang-Huy Duong[c], Thu-Lan Dam[c,d], Lukáš Ševčík[e], Dominik Uhrin[e], Miroslav Voznak[e]

[a] *School of Natural Sciences and Humanities, Harbin Institute of Technology Shenzhen Graduate School, Shenzhen, China*

[b] *School of Computer Science and Technology Harbin Institute of Technology Shenzhen Graduate School, Shenzhen, China*

[c] *College of Computer Science and Electronic Engineering, Hunan University, China*

[d] *Faculty of Information Technology, Hanoi University of Industry, Vietnam*

[e] *Department of Telecommunications, Faculty of Electrical Engineering and Computer Science, VSB-Technical University of Ostrava, 17. listopadu 15, 708 00 Ostrava-Poruba, Czech Republic*

Introduction

- **Frequent itemset mining is** a popular data mining task.
- It consists of discovering sets of items (itemsets) frequently appearing in customer transactions.
- Many applications in various domains.

Example: {tomato, wine, cheese} may be a frequent itemsets in transactions of a retail store (a set of items bought by many customers)

Def.: Support

A transaction database

Transaction	item
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}

Support of an itemset: the number of transactions where it appears

Example:
{a,c} has a support of **4**

Discovering Frequent Patterns

- **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 $\boldsymbol{2^n - 1}$ 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** $\subset$ **Y**, then the support of Y is less than or equal to the support of **X**.

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}

Example

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**
  - It is not designed to find patterns that are periodic.

Introduction (cont'd)

Periodic Frequent Pattern Mining: discovering groups of items that appear periodically in a sequence of transactions.

Several applications.

Example:

{baguette, cheese, orange juice} may be a *frequent periodic pattern* for a particular customer, occurring every week.

Introduction (cont'd)

Previous work:

- a pattern is generally considered « **periodic** » if it has no period greater than a maximum periodicity threshold, set by the user.

Problem:

- As a result, a pattern can be discarded if it has a single period greater than the threshold. But this is too strict.

Our solution:

- We introduce novel periodicity measures: the **average periodicity** and **minimum periodicity**
- We present an efficient algorithm: **PFPM**

Def.: Periodicity

A transaction database

Transaction	item
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}

1 2 2 1 1

Periods of an itemset:
the number of transactions between each occurrence of the itemset

Example:
The periods of the itemset **{a,c}** are: **1,2,2,1,1**

Def.: Periodicity

A transaction database

Transaction	item	
T_1	{**a, c**}	1
T_2	{e}	2
T_3	{**a**,b,**c**,d,e}	
T_4	{b,c,d,e}	2
T_5	{**a,c**,d}	
T_6	{**a,c,**e}	1
T_7	{b,c,e}	1

Periods of an itemset:
the number of transactions between each occurrence of the itemset

Example:
The periods of the itemset **{a,c}** are: **1,2,2,1,1**

The **maximum periodicity of {a,c}** is **2**

Periodic frequent pattern : a pattern that has a support and maximum periodicty not greater than two threshold set by the user named **minsup** and **maxper**.

Example

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

This paper proposes two new measures: the **minimum periodicity** and the **average periodicity**

New problem definition

- **Goal:** find each pattern X such that
  - $minAvg \leq avgper(X) \leq maxAvg$
  - $Minper(X) \geq minPer$
  - $Maxper(X) \leq maxper$

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

  The new problem definition gives more control to the users by considering the average periodicity and minimum periodicity.

The PFPM algorithm

We don't want to test all the possible itemsets. The search space is huge. We thus propose three new lemmas and two new theorem for pruning the search space using the new measures.

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 $maxper(Y) \geq maxper(X)$ [9].

Theorem 1 (Maximum periodicity pruning). Let X be an itemset appearing in a database D. X and its supersets are not PFPs if $maxper(X) > maxPer$. Thus, if this condition is met, the search space consisting of X and all its supersets can be discarded. (this follows from Lemma 4).

Theorem 2 (Average periodicity pruning). Let X be an itemset appearing in a database D. X is not a PFP as well as all of its supersets if $avgper(X) > maxAvg$, or equivalently if $|g(X)| < (|D|/maxAvg) - 1$. Thus, if this condition is met, the search space consisting of X and all its supersets can be discarded.

The PFPM algorithm (cont'd)

- PFPM: a novel algorithm
- It performs a depth-first search.
  - It starts with single items {a}, {b}, {c}, {d}…
  - Then, the algorithm recursively appends items to to generate larger itemsets:

    {a,b}, {a,b,c}, {a,b,c,d}…. {a,b,d}…

To reduce the search space, the pruning Theorems 1 and 2 are applied.

The PFPM algorithm (cont'd)

- The algorithm is inpsired by the Eclat algorithm for frequent itemset mining.
- PFPM 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
- This allows quickly calculating the periods of any itemset (*see the paper for details*)

Pseudocode

Algorithm 1: The PFPM algorithm

input : a transaction database D, $minAvg$, $maxAvg$, $minPer$ and $maxPer$
output: the set of periodic itemsets

```
1 Scan D once to calculate minper({i}), maxper({i}), and s({i}) for each item i ∈ I;
2 γ ← (|D|/maxAvg) − 1;
3 I* ← each item i such that s({i}) ≥ γ and maxper({i}) ≤ maxPer;
4 Let ≻ be the total order of support ascending values on I*;
5 Scan D to build the tid-list of each item i ∈ I*;
6 PFPMSearch ( I*, γ, minAvg, minPer, maxPer, |D|);
```

Algorithm 2: The *PFPMSearch* procedure

input : *ExtensionsOfP*: a set of extensions of an itemset P, γ, $minAvg$, $minPer$, $maxPer$, $|D|$
output: the set of periodic frequent itemsets

```
1  foreach itemset Px ∈ ExtensionsOfP do
2      avgperPx ← |D|/(|Px.tidlist| + 1);
3      if minAvg ≤ avgperPx ≤ maxAvg ∧ Px.tidlist.minp ≥ minPer ∧ Px.tidlist.maxp ≤ maxPer
         then output Px;
4      if avgperPx ≥ γ and Px.tidlist.maxp ≤ maxPer then
5          ExtensionsOfPx ← ∅;
6          foreach itemset Py ∈ ExtensionsOfP such that y ≻ x do
7              Pxy ← Px ∪ Py;
8              Pxy.tidlist ← Intersect (Px, Py);
9              ExtensionsOfPx ← ExtensionsOfPx ∪ {Pxy};
10         end
11         PFPMSearch (ExtensionsOfPx, γ, minAvg, minPer, maxPer, |D|);
12     end
13 end
```

Algorithm 3: The Intersect procedure

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

```
1 TidListOfPxy ← ∅;
2 foreach Tid v ∈ Px.tidlist such that v ∈ Py.tidlist do
3     period_v ← calculatePeriod(v,TidListOfPxy);
4     UpdateMinPerMaxPer(TidListOfPxy, period_v);
5     TidListOfPxy ← TidListOfPxy ∪ {v};
6 end
7 return TidListOfPxy;
```

Experimental Evaluation

Datasets' characterictics

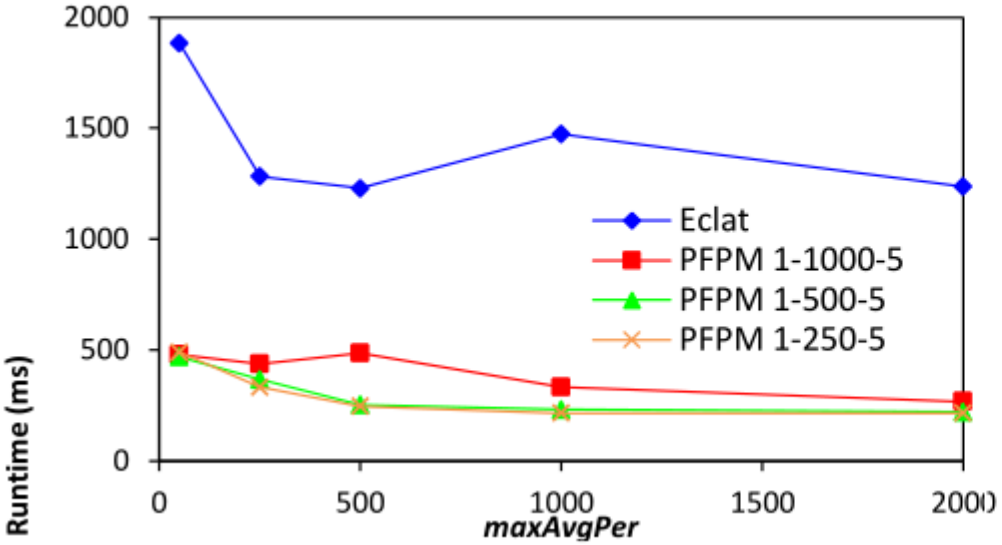
Dataset	transaction count	distinct item count	average transaction length
Retail	88,162	16,470	10.30
Chainstore	1,112,949	46,086	7.26
Mushroom	8,124	120	23
Foodmart	1,559	4,141	4.4

- We compared PFPM with the popular ECLAT algorithm for frequent itemset mining.
- Eclat find all frequent itemsets (does not eliminate non periodic patterns).
- We varied the various parameters of PFPM
- Java, Windows 7, 5 GB of RAM

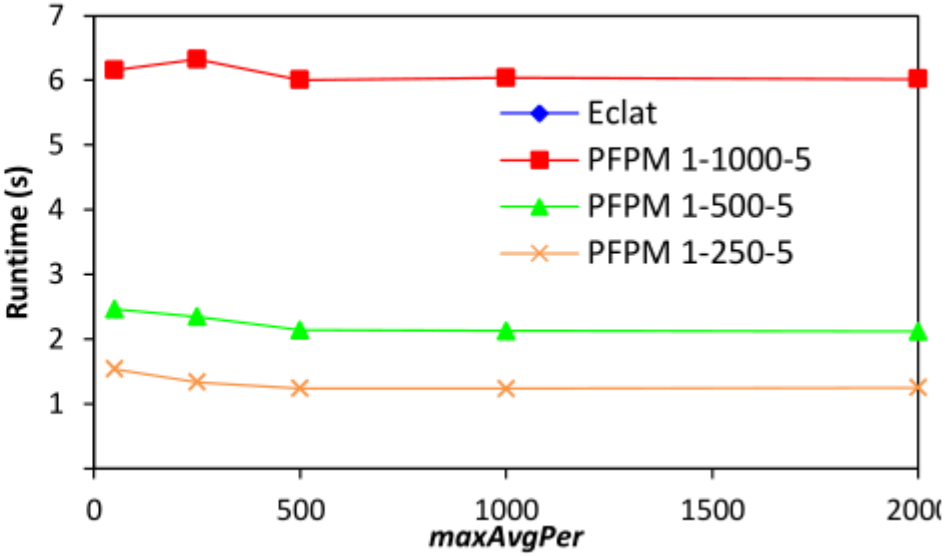
Execution times

PFPM VW-X represents the PFPM algorithm with **minPer = V** , **maxPer = W**, and **minAvg = X.**

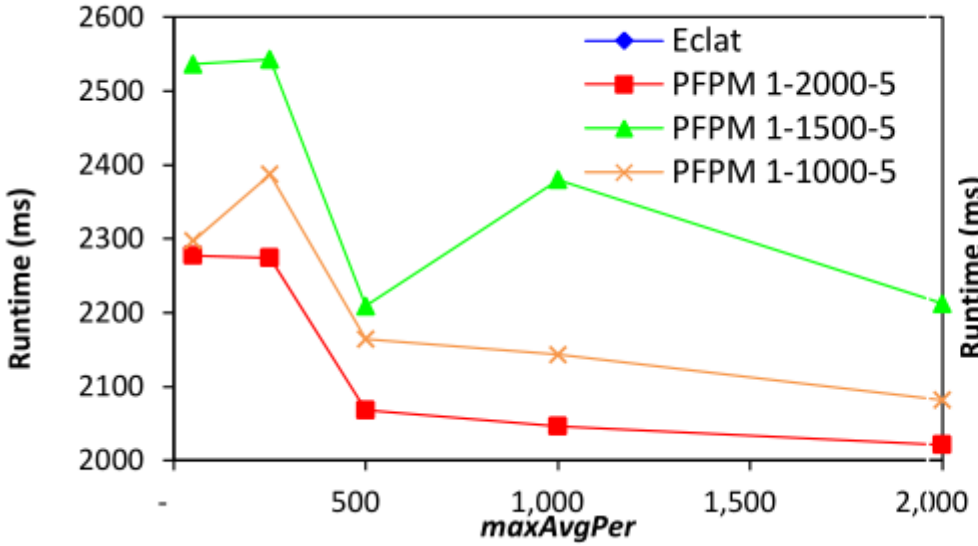
Retail



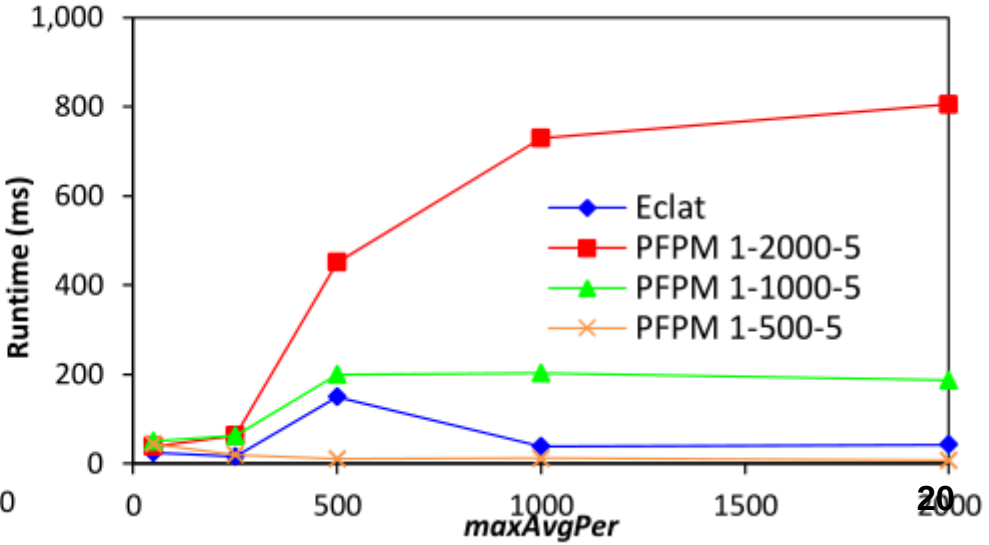
Mushroom



Chainstore



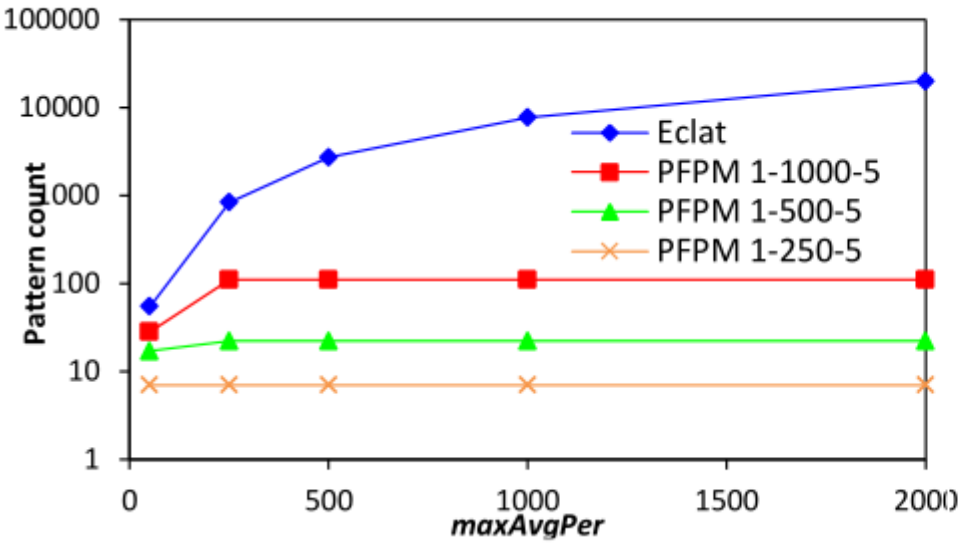
Foodmart



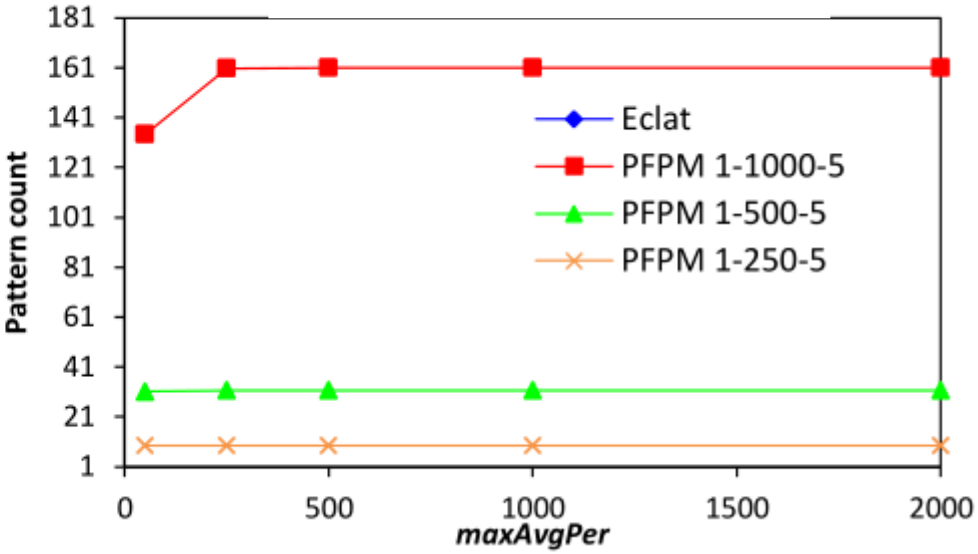
PFPM can be much faster than Eclat because it filters many non periodic patterns

Number of Pattern found

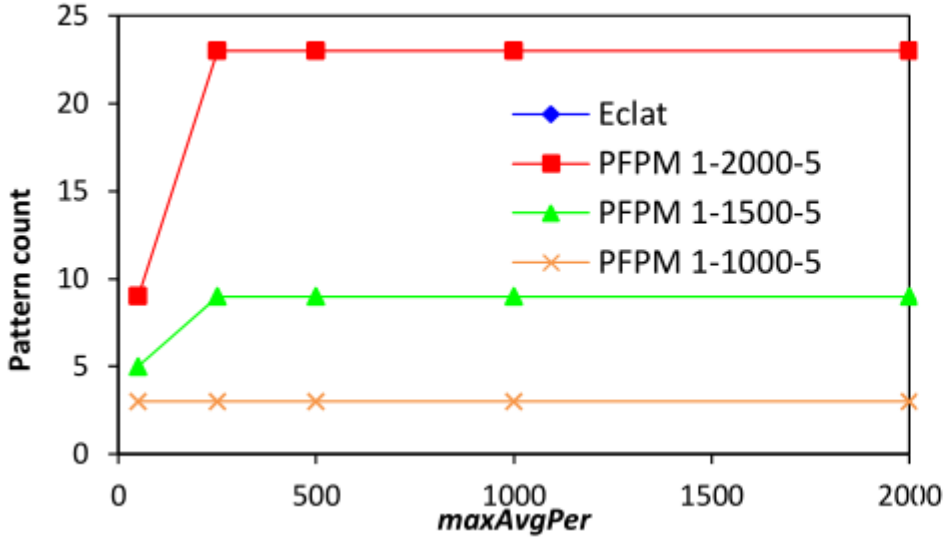
Retail



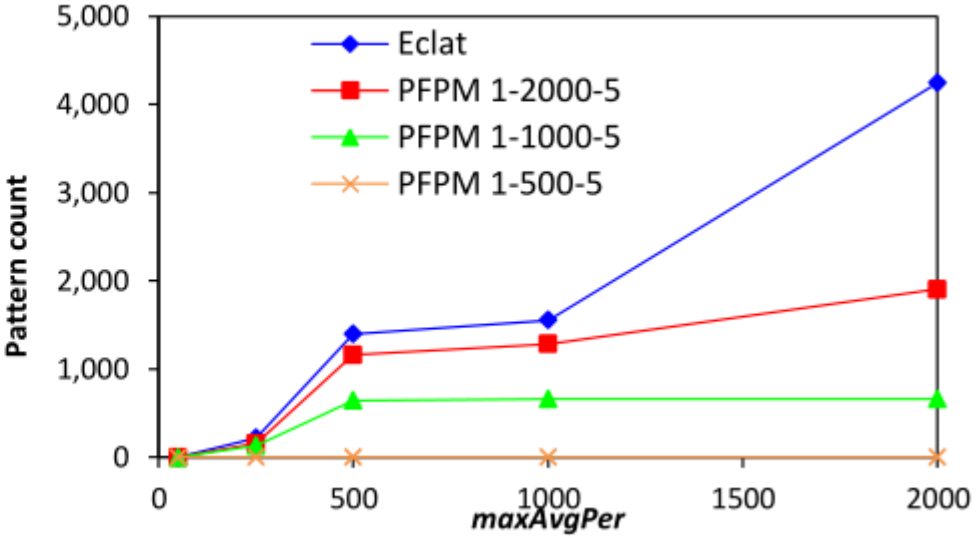
Mushroom



Chainstore



Foodmart



As seen in this experiment, the PFPM algorithm can filter 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.
- PFPM can use up to four and five times less **memory** then Eclat.
  - For example, on retail and maxAvg = 2,000, Eclat and PFPM 1-5000-5-500 respectively consumes 900 MB and 189 MB of memory.

Conclusion

- PFPM: A novel algorithm for mining periodic frequent patterns.
- Our proposal:
  - Two new periodicity measure: average and minimum periodicity
  - A novel algorithm named PFPM
- Experimental results:
  - Can eliminate a large number of non-periodic patterns.
  - Is much faster than Eclat in many cases.
- Source code and datasets available as part of the **SPMF data mining library (**GPL 3).

Open source Java data mining software, 66 algorithms
http://www.phillippe-fournier-viger.com/spmf/

Thank you. Questions?

Open source Java data mining software, 110 algorithms
http://www.phillippe-fournier-viger.com/spmf/