# RULEGROWTH: Mining Sequential Rules Common to Several Sequences by Pattern-Growth

**Philippe Fournier-Viger**[1]

Roger Nkambou[2]

Vincent Shin-Mu Tseng[1]

[1]**National Cheng Kung University (Taiwan)**

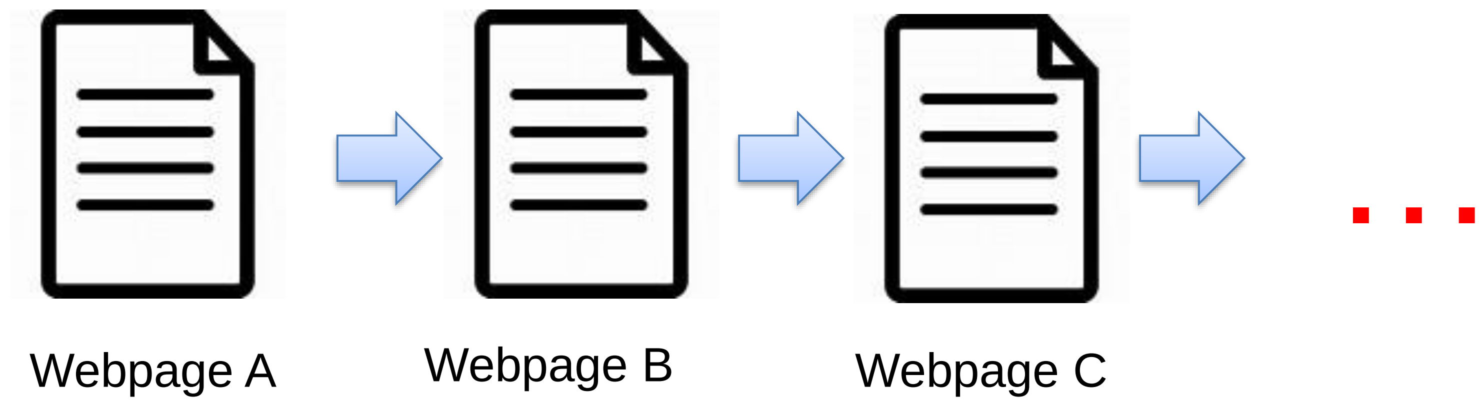
[2]University of Québec in Montréal (Canada)

Fournier-Viger, P., Wu, C.-W., Tseng, V.S., Cao, L., Nkambou, R. (2015). **Mining Partially-Ordered Sequential Rules Common to Multiple Sequences**. IEEE Transactions on Knowledge and Data Engineering (TKDE), 27(8): 2203-2216.

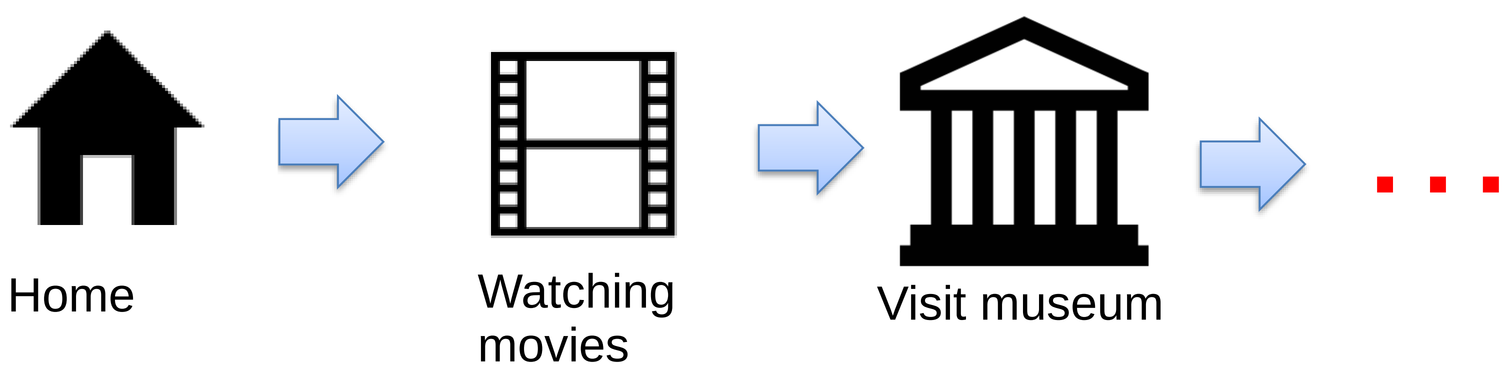
# Introduction

- Many **databases** contain large amount of **temporal or sequential information.**
- It is a challenge to develop algorithms for discovering useful patterns in these databases.
- Different kind of **temporal** patterns: repetitive patterns, trends, similar patterns, sequential patterns, etc.
- In this paper, we are interested by **sequence databases** containing sequences of **discrete events or symbols .**

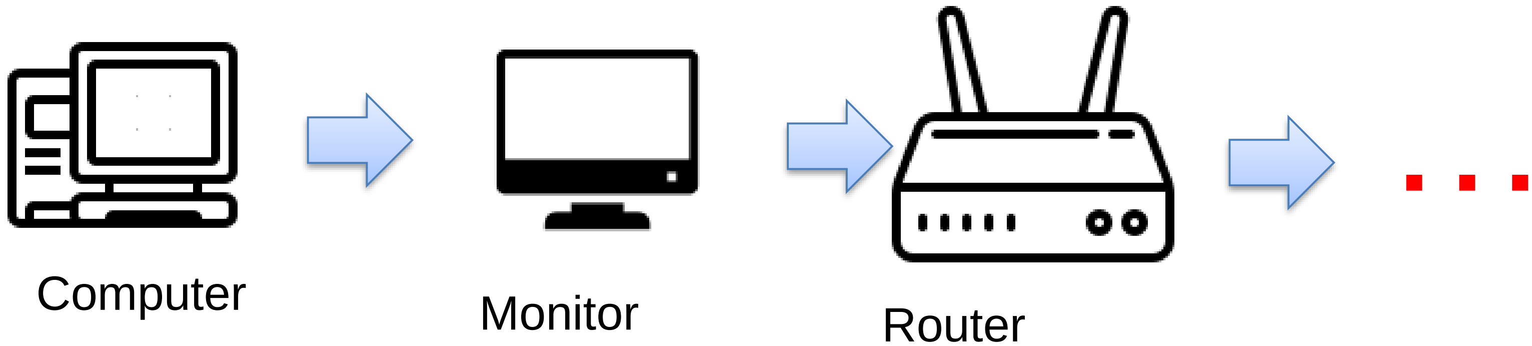
# Sequences of webpage clicks




# Sequences of activities



# Sequences of purchases




# Sequences of words

**Where** **are** **you** going?

# Sequence Database

- Let there be a set of symbols (**e.g.** *a, b, c, d… g*) called **items**.
- An **itemset** is a set of **items** that appeared simultaneously.
- Each **sequence** is an ordered list of itemsets.

| ID | Sequences |
|---|---|
| *seq1* | $\{a, b\}, \{c\}, \{f\}, \{g\}, \{e\}$ |
| *seq2* | $\{a, d\}, \{c\}, \{b\}, \{a, b, e, f\}$ |
| *seq3* | $\{a\}, \{b\}, \{f\}, \{e\}$ |
| *seq4* | $\{b\}, \{f, g\}$ |

# Sequential pattern mining

## Input:

– A sequence database (a set of sequences)

– A ***minsup*** threshold

## Output:

– All sub-sequences having a support greater or equal to ***minsup***.

**Example**: minsup = 50 % (2 sequences)

**A sequence database**

| IFD | sequence |
|---|---|
| 1 | <{a}, {a,b,c} {a, c} {d} {c, f}> |
| 2 | <{a, d}, {c} {b, c} {a, e}> |
| 3 | <{e, f}, {a, b} {d, f} {c}, {b}> |
| 4 | <{e}, {g}, {a, f} {c} {b}, {c}> |

**Sequential patterns**

| Pattern | support |
|---|---|
| {a} | 100 % |
| <{a}, {b,c} > | 50 % |
| <{a, b} > | 50 % |
| … | … |

*Fournier-Viger, P., Lin, J. C.-W., Kiran, R. U., Koh, Y. S., Thomas, R. (2017). A Survey of Sequential Pattern Mining. Data Science and Pattern Recognition (DSPR), vol. 1(1), pp. 54-77.*

# Limitation of sequential pattern mining

- SPM is not very useful for making **predictions.**
- For example, consider the pattern **{x},{y}.**

| IFD | sequence |
|---|---|
| 1 | <{x}, {w}, {z}, {y}> |
| 2 | <{x}, {z}, {z}> |
| 3 | <{x}, {z}, {y}> |
| 4 | <{x}, {z}, {z}> |

- Although ***y*** appears frequently after ***x,*** there are also many cases where ***x*** is **not** followed by ***y*.**
- If we want to make **predictions**, we need a measurement of the confidence that if ***x*** occurs, it will be followed by ***y***.

# A solution: sequential Rule Mining

- **Sequential rules**: a type of sequential patterns that incorporate a measure of **confidence**.
- A **sequential rule** typically has the form X→Y and has a *confidence* and a *support.*

  {*bread, milk*} → {*coffee*}  confidence : 75 %

- **Several algorithms** by Manila et al. (1997), Hamilton & Karimi (2005), Hsieh (2006), Deogun (2005). But mostly for discovering rules in a **single sequence**.
- In this paper, we are interested by finding rules appearing in **multiple sequences**.

# Finding sequential rule in multiple sequences?

- Zaki et al. (2001) proposed the **RuleGen** algorithm
- Rules of the form X → Y where X and Y are **sequential patterns**.

**A sequence database**

1: {*Vivaldi*}, {*Mozart*}, {*Handel*}, {*Berlioz*}
2: {*Mozart*}, {Bach},{*Paganini*}, {*Vivaldi*}, {*Handel*}, {*Berlioz*}
3: {*Handel*}, {*Vivaldi*}, {*Mozart*}, {*Ravel*}, {*Berlioz*}
4: {*Vivaldi*}, {*Mozart*}, {*Handel*}, {*Bach*}, {*Berlioz*}
5: {*Mozart*}, {*Bach*}, {*Vivaldi*}, {*Handel*}
6: {*Vivaldi*}, {*Handel*}, {*Mozart*}, {*Bach*}

**Some sequential rules**

R1: {*Vivaldi*}, {*Mozart*}, {*Handel*} ⇒ {*Berlioz*
R2: {*Mozart*}, {*Vivaldi*}, {*Handel*} ⇒ {*Berlioz*},
R3: {*Handel*}, {*Vivaldi*}, {*Mozart*} ⇒ {*Berlioz*},
R4: {*Handel, Vivaldi*}, {*Mozart*} ⇒ {*Berlioz*},
R5: {*Handel*}, {*Vivaldi, Mozart*} ⇒ {*Berlioz*},
R6: {*Handel, Vivaldi, Mozart*} ⇒ {*Berlioz*}.

M. J. Zaki, “SPADE: An Efficient Algorithm for Mining Frequent Sequences,”Machine Learning, vol. 42, no.1-2, pp. 31-60, 2001.

# Finding sequential rule in multiple sequences?

- Zaki et al. (2001) proposed the **RuleGen** algorithm
- Rules of the form X → Y where X and Y are **sequential patterns**.

**A sequence database**

1: {*Vivaldi*}, {*Mozart*}, {*Handel*}, {*Berlioz*}
2: {*Mozart*}, {Bach},{*Paganini*}, {*Vivaldi*}, {*Handel*}, {*Berlioz*}
3: {*Handel*}, {*Vivaldi*}, {*Mozart*}, {*Ravel*}, {*Berlioz*}
4: {*Vivaldi*}, {*Mozart*}, {*Handel*}, {*Bach*}, {*Berlioz*}
5: {*Mozart*}, {*Bach*}, {*Vivaldi*}, {*Handel*}
6: {*Vivaldi*}, {*Handel*}, {*Mozart*}, {*Bach*}

**Some sequential rules**

R1: {*Vivaldi*}, {*Mozart*}, {*Handel*} ⇒ {*Berlioz*
R2: {*Mozart*}, {*Vivaldi*}, {*Handel*} ⇒ {*Berlioz*},
R3: {*Handel*}, {*Vivaldi*}, {*Mozart*} ⇒ {*Berlioz*},
R4: {*Handel, Vivaldi*}, {*Mozart*} ⇒ {*Berlioz*},
R5: {*Handel*}, {*Vivaldi, Mozart*} ⇒ {*Berlioz*},
R6: {*Handel, Vivaldi, Mozart*} ⇒ {*Berlioz*}.

**Problem**: all these rules are very similar!!!... There are 23 such rules with these items.

**R1** support: 33% confidence: 100%
**R2** support: 16%, confidence: 50%
**R3** support: 16%, confidence: 100%
…

# **Our solution**: partially-ordered sequential rules

**Some sequential rules**

R1: {*Vivaldi*}, {*Mozart*}, {*Handel*} ⇒ {*Berlioz*
R2: {*Mozart*}, {*Vivaldi*}, {*Handel*} ⇒ {*Berlioz*},
R3: {*Handel*}, {*Vivaldi*}, {*Mozart*} ⇒ {*Berlioz*},
R4: {*Handel, Vivaldi*}, {*Mozart*} ⇒ {*Berlioz*},
R5: {*Handel*}, {*Vivaldi, Mozart*} ⇒ {*Berlioz*},
R6: {*Handel, Vivaldi, Mozart*} ⇒ {*Berlioz*}.

By removing the order on the left side or right side of a rule, we can obtain a single rule:

{*Mozart, Vivaldi, Handel*} ⇒ {*Berlioz*}

support: 75% confidence: 66%

*P. Fournier-Viger, U. Faghihi, R. Nkambou and E. Mephu Nguifo, "CMRules: An Efficient Algorithm for Mining Sequential Rules Common to Several Sequences," Knowledge Based Systems, vol. 25, no. 1, pp. 63-76, 2012.*

# **Our solution**: partially-ordered sequential rules

- A **sequential rule** $X \Rightarrow Y$ is a relationship between two disjoint and non empty itemsets X,Y.
- A sequential rule $X \Rightarrow Y$ has **two properties**:
  - **Support:** the number of sequences where X occurs before Y, divided by the number of sequences.
  - **Confidence** the number of sequences where X occurs before Y, divided by the number of sequences where X occurs.
- **The task**: finding all **valid rules**, rules with a support and confidence not less than user-defined thresholds *minSup* and *minConf* (Fournier-Viger, 2010).

# An example of Sequential Rule Mining

Consider *minSup*= 0.5 and *minConf*= 0.5:

| ID | Sequences |
|---|---|
| *seq1* | {*a*, *b*},{*c*},{*f*},{*g*},{*e*} |
| *seq2* | {*a*, *d*},{*c*},{*b*},{*a*, *b*, *e*, *f*} |
| *seq3* | {*a*},{*b*},{*f*},{*e*} |
| *seq4* | {*b*},{*f*, *g*} |

A sequence database

→

| ID | Rule | Support | Confidence |
|---|---|---|---|
| r1 | {*a*, *b*, *c*}⇒{*e*} | 0.5 | 1.0 |
| r2 | {*a*}→{*c*, *e*, *f*} | 0.5 | 0.66 |
| r3 | {*a*, *b*} →{*e*, *f*} | 0.5 | 1.0 |
| r4 | {*b*} →{*e*, *f*} | 0.75 | 0.75 |
| r5 | {*a*} →{*e*, *f*} | 0.75 | 1.0 |
| r6 | {*c*} →{*f*} | 0.5 | 1.0 |
| r7 | {*a*}→{*b*} | 0.5 | 0.66 |
| … | … | … | … |

Some rules found

# Previous Algorithms

- **CMRules (2010)**: An association rule mining based algorithm for the discovery of sequential rules.
- **CMDeo (2010)**: An Apriori based algorithm for the discovery of sequential rules.
- **Limitation**: Both algorithms use a « generate-candidate-and-test » approach that may generate a large amount of candidates for dense datasets. Many candidates do not appear in the database.

# RuleGrowth

**An algorithm inspired by PrefixSpan. It generates rules by growing them one item at a time.**

**The input is:**

– A sequence database :

| ID | Sequences |
|---|---|
| *seq1* | $\{a, b\},\{c\},\{f\},\{g\},\{e\}$ |
| *seq2* | $\{a, d\},\{c\},\{b\},\{a, b, e, f\}$ |
| *seq3* | $\{a\},\{b\},\{f\},\{e\}$ |
| *seq4* | $\{b\},\{f, g\}$ |

– minsup = 0.5 %,

– minconf = 0.5%

# RuleGrowth

**Step1:** Scan database to calculate the support of each item. Keep only frequent items.

| ID | Sequences |
|---|---|
| $seq1$ | $\{a, b\},\{c\},\{f\},\{g\},\{e\}$ |
| $seq2$ | $\{a, d\},\{c\},\{b\},\{a, b, e, f\}$ |
| $seq3$ | $\{a\},\{b\},\{f\},\{e\}$ |
| $seq4$ | $\{b\},\{f, g\}$ |

**Frequent items:**

| Item | support |
|---|---|
| a | 75 % |
| b | 100 % |
| c | 50 % |
| ~~d~~ | ~~25 %~~ |
| e | 75 % |
| f | 100 % |

# RuleGrowth

**Step2:** For each pairs of frequent items, try to create a rule with only two items. **e.g. {a}⇒{b}.**

| ID | Sequences |
|---|---|
| *seq1* | {*a*, *b*},{*c*},{*f*},{*g*},{*e*} |
| *seq2* | {*a*, *d*},{*c*},{*b*},{*a*, *b*, *e*, *f*} |
| *seq3* | {*a*},{*b*},{*f*},{*e*} |
| *seq4* | {*b*},{*f*, *g*} |

**Frequent items:**

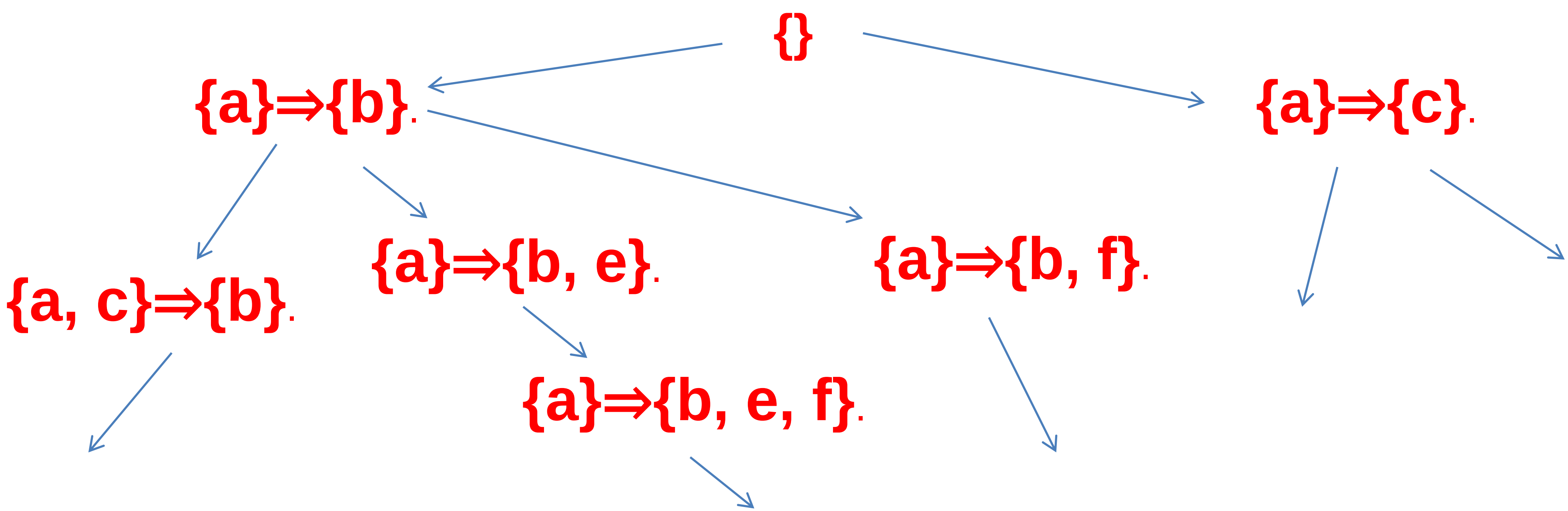
| Item | support |
|---|---|
| a | 75 % |
| b | 100 % |
| c | 50 % |
| ~~d~~ | ~~25 %~~ |
| e | 75 % |
| f | 100 % |

For each rule, calculate the confidence and support. If the confidence and support are respectively higher or equal to minconf and minsup, the rule is output.

**{a}⇒{b}. Support = 50 % Confidence = 2/3**

# RuleGrowth

**Step3:** Find larger rules by recursively scanning the datatabase for adding a single item at a time to the left or right part of each rule (these processes are called *left* and *right expansions*).




**For each rule, calculate the confidence and support. If the confidence and support are respectively higher or equal to minconf and minsup, output the rule.**

# RuleGrowth

- **When a rule should be expanded?**
  A rule should be expanded only if it has the minimum support.

# RuleGrowth

**How to choose items for performing left expansions of a rule $X \Rightarrow Y$ ?**

Scan the sequences containing the rule and note items appearing in at least *minsup* × |S| sequences before the last occurrence of Y.

For example: **{a}⇒{b}**

| ID | Sequences |
|---|---|
| *seq1* | $\{a, b\}, \{c\}, \{f\}, \{g\}, \{e\}$ |
| *seq2* | $\{a, d\}, \{c\}, \{b\}, \{a, b, e, f\}$ |
| *seq3* | $\{a\}, \{b\}, \{f\}, \{e\}$ |
| *seq4* | $\{b\}, \{f, g\}$ |

In this example, no item can expand the left itemset of the rule!

# RuleGrowth

**How to choose items for performing right expansions of a rule X⇒Y ?**

Scan the sequences containing the rule and note items appearing in at least *minsup* x |S| sequences after the first occurrence of X.

For example: **{a}⇒{b}**

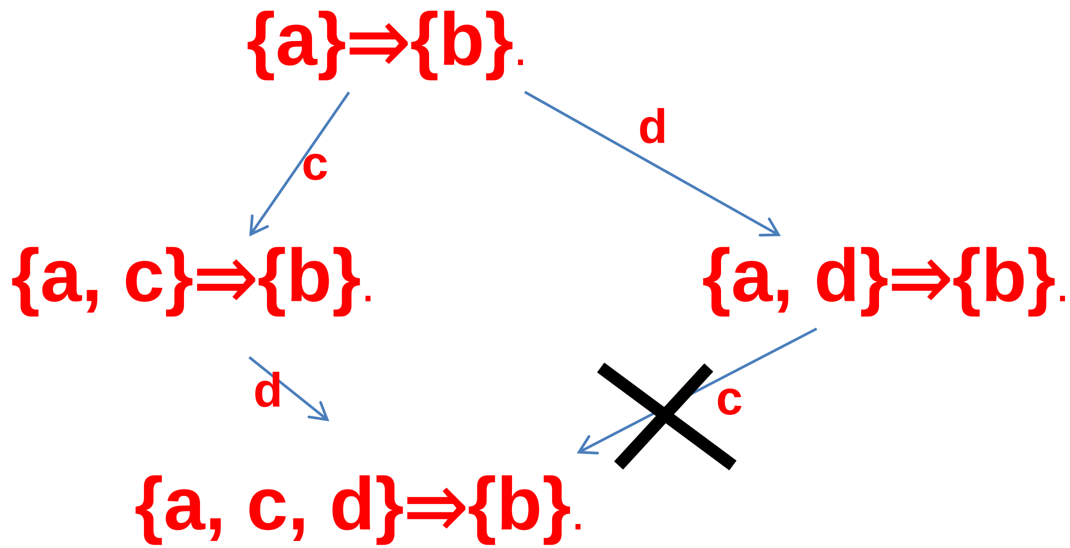
| ID | Sequences |
|---|---|
| *seq1* | {*a*, *b*},{*c*},{*f*},{*g*},{*e*} |
| *seq2* | {*a*, *d*},{*c*},{*b*},{*a*, *b*, *e*, *f*} |
| *seq3* | {*a*},{*b*},{*f*},{*e*} |
| *seq4* | {*b*},{*f*, *g*} |

The following items meet these criteria:

c : seq1, seq2
e : seq1, seq2, seq3
f : seq1, seq2, seq3

# How to avoid generating the same rules twice?

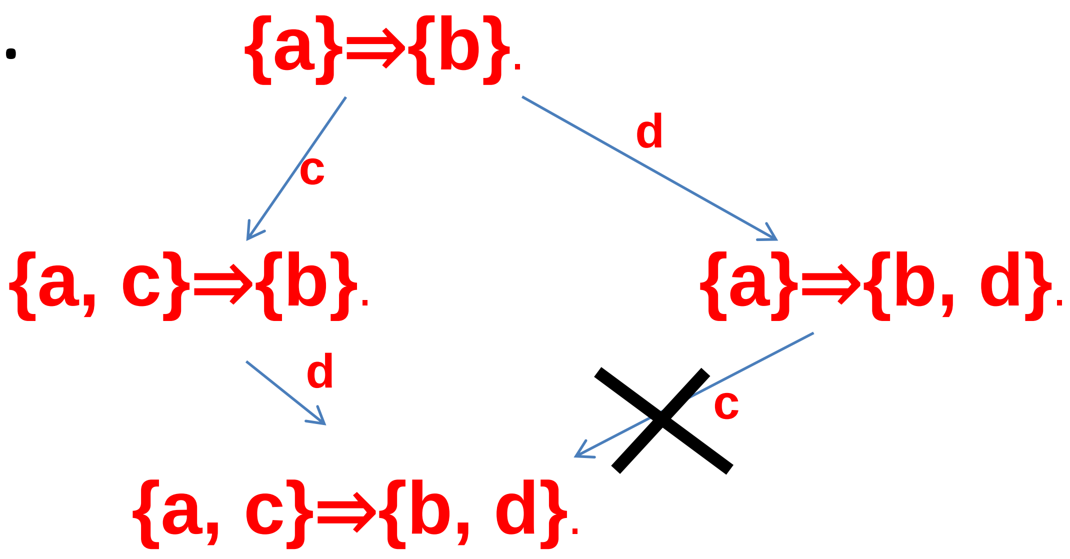
**Problem 1:** The same rule can be generated by adding items in different orders.




**Solution:** Add only an item to the left/right part of a rule if the item is larger than all items already in the left/right part.

# How to avoid generating the same rules twice?

**Problem 2:** the same rule can be generated by adding items in different orders of left/right expansions.




**Solution:** Do not allow performing a left expansion after a right expansion. But allow performing a right expansion after a left expansion.

# Implementation

**Optimization 1**:The set of sequences containing X, Y, and X⇒Y is kept for each rule X⇒Y generated so that the confidence can be calculated efficiently.

**Optimization 2**: During the first database scan, record the first and last occurrence of each item for each sequence.

- This allows to create initial rules very efficiently.
- This allows to avoid scanning sequences completely when searching for items for expansions.

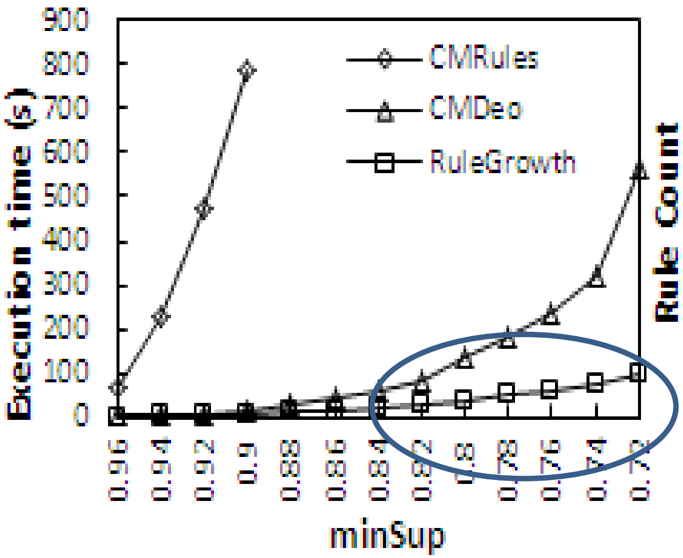
# Performance Evaluation

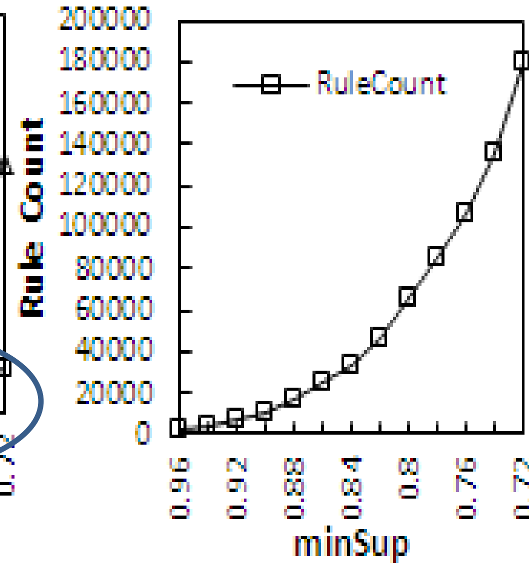
- RuleGrowth, CMRules and CMDEO.
- Java, 1GB of RAM
- Three real-life public datasets.

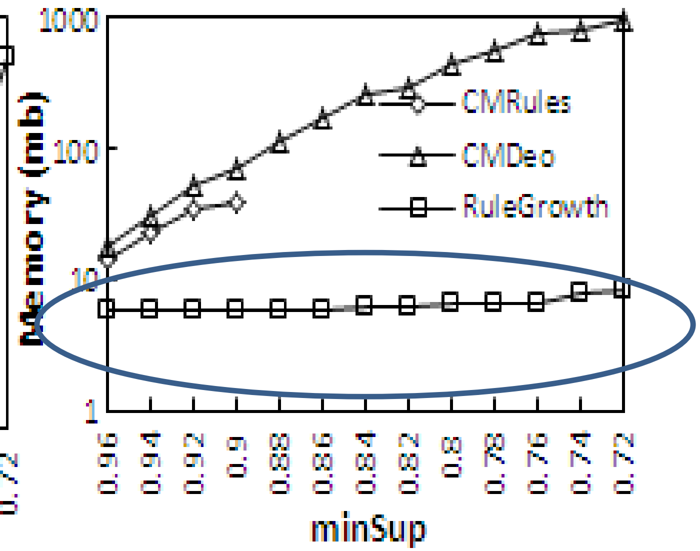
| | **Kosarak** | **BMS** | **Toxin-Snake** |
|---|---|---|---|
| Sequence count | 70,000 | 59,601 | 163 |
| Item count | 21,144 | 497 | 20 |
| Average item count by sequence | 7.97 | 2.51 | 60.61 |
| Average different item count by sequence | 7.97 | 2.51 | 17.84 |

# Influence of *minsup*

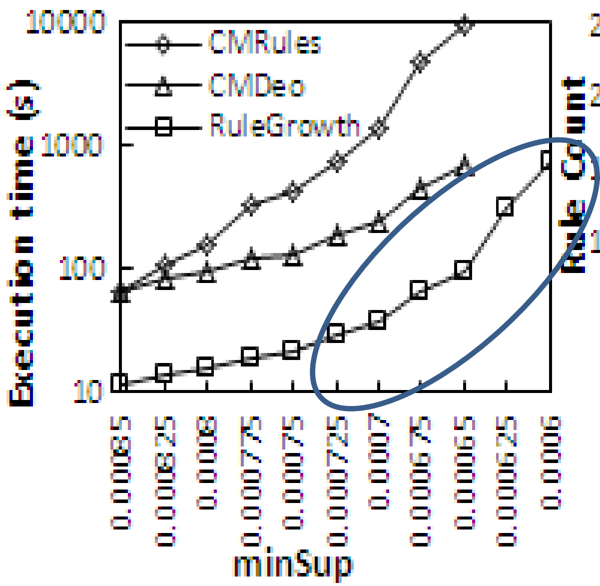
**Snake**

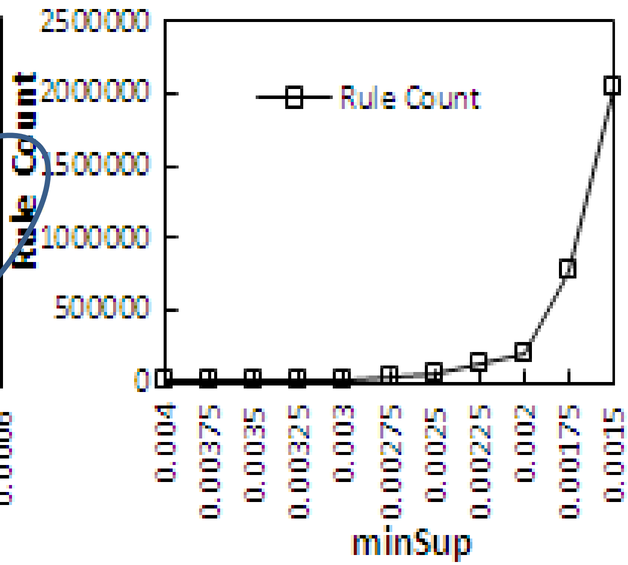


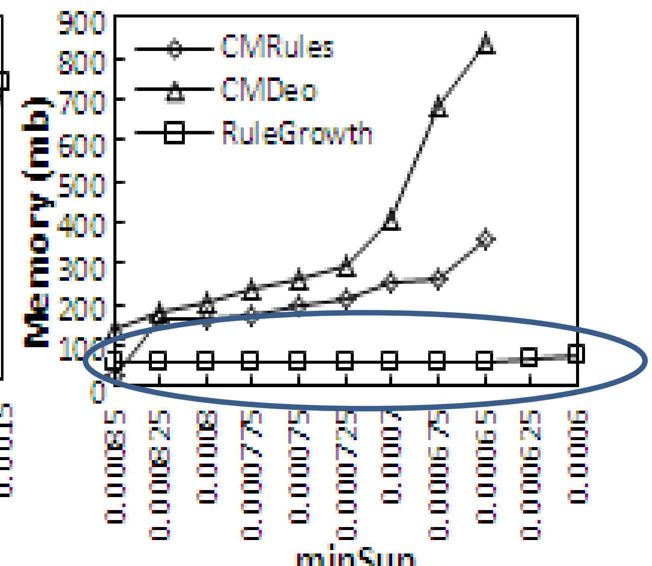




**BMS**

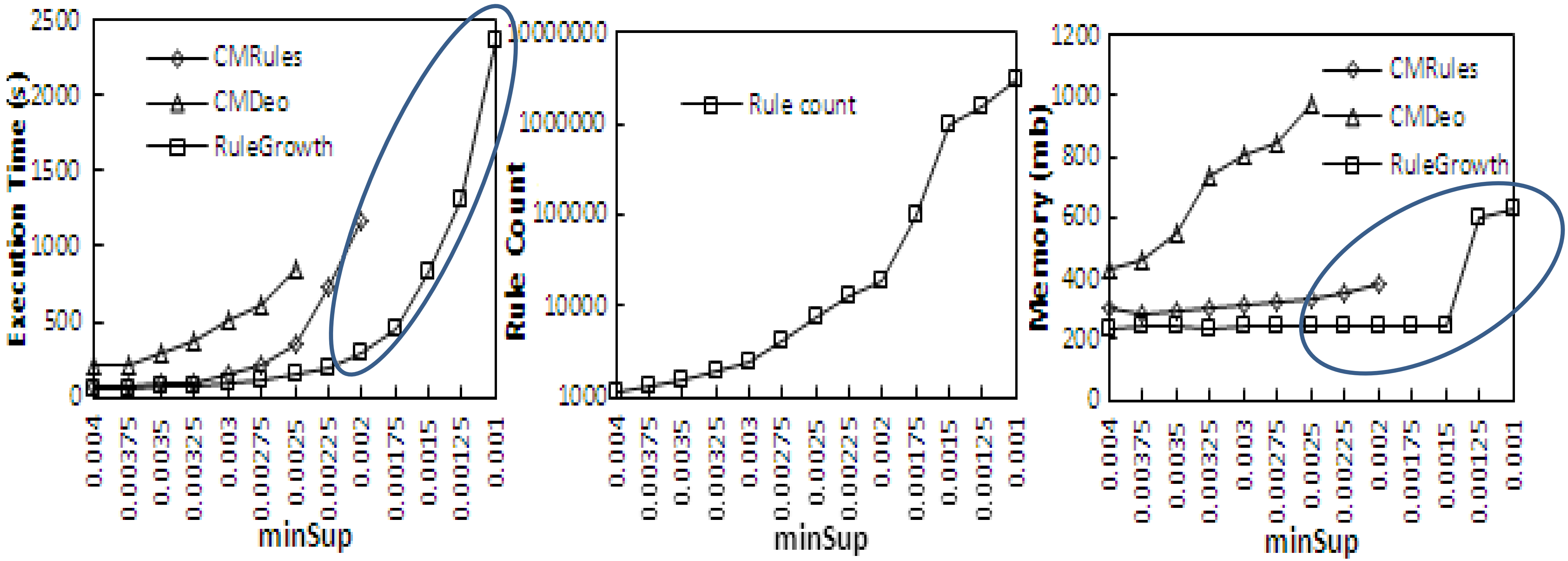


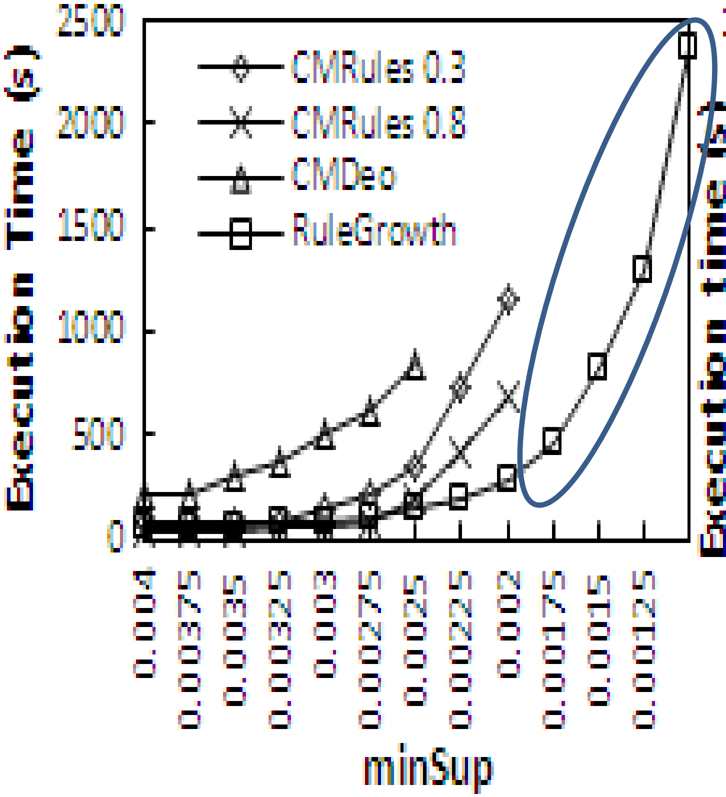
# Influence of *minsup*

**Kosarak**

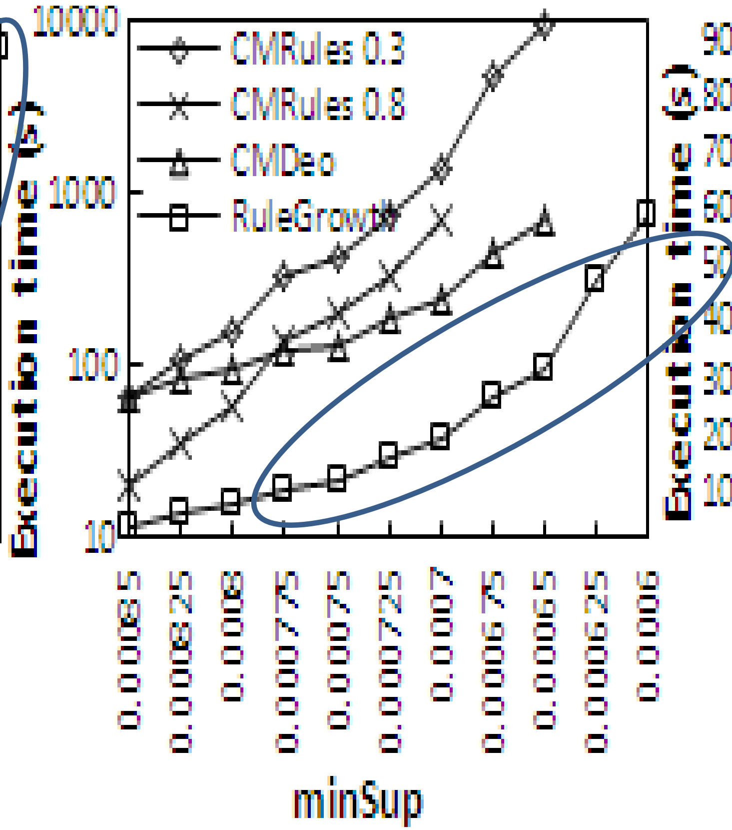


# Influence of *minconf*

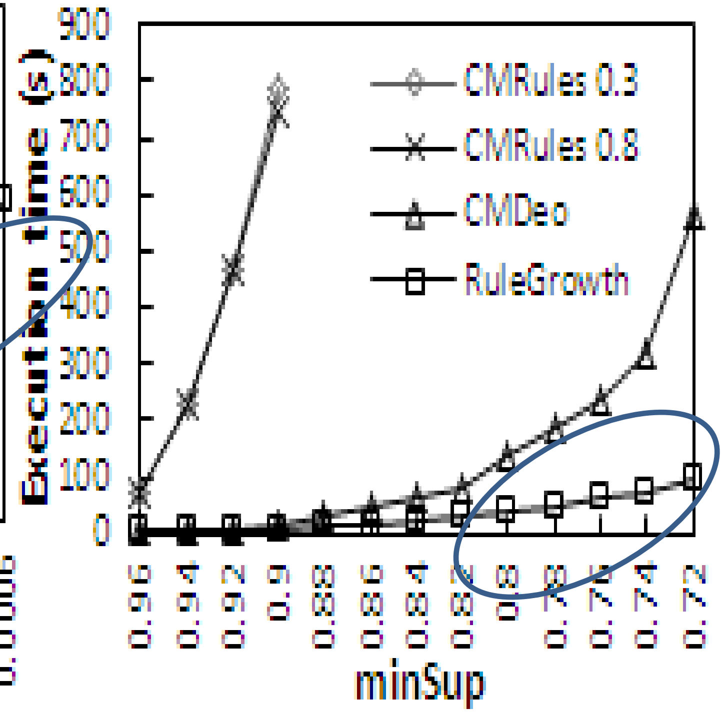
**Kosarak**




**BMS**




**Snake**



# Conclusion

**RuleGrowth**,

- Is a novel algorithm for mining sequential rules common to several sequences,
- It outperforms **CMRules** and **CMDeo** in terms of execution time and memory usage.
- Source code and datasets available as part of the **SPMF data mining library (**GPL 3).

**Open source Java data mining software**, 150 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.

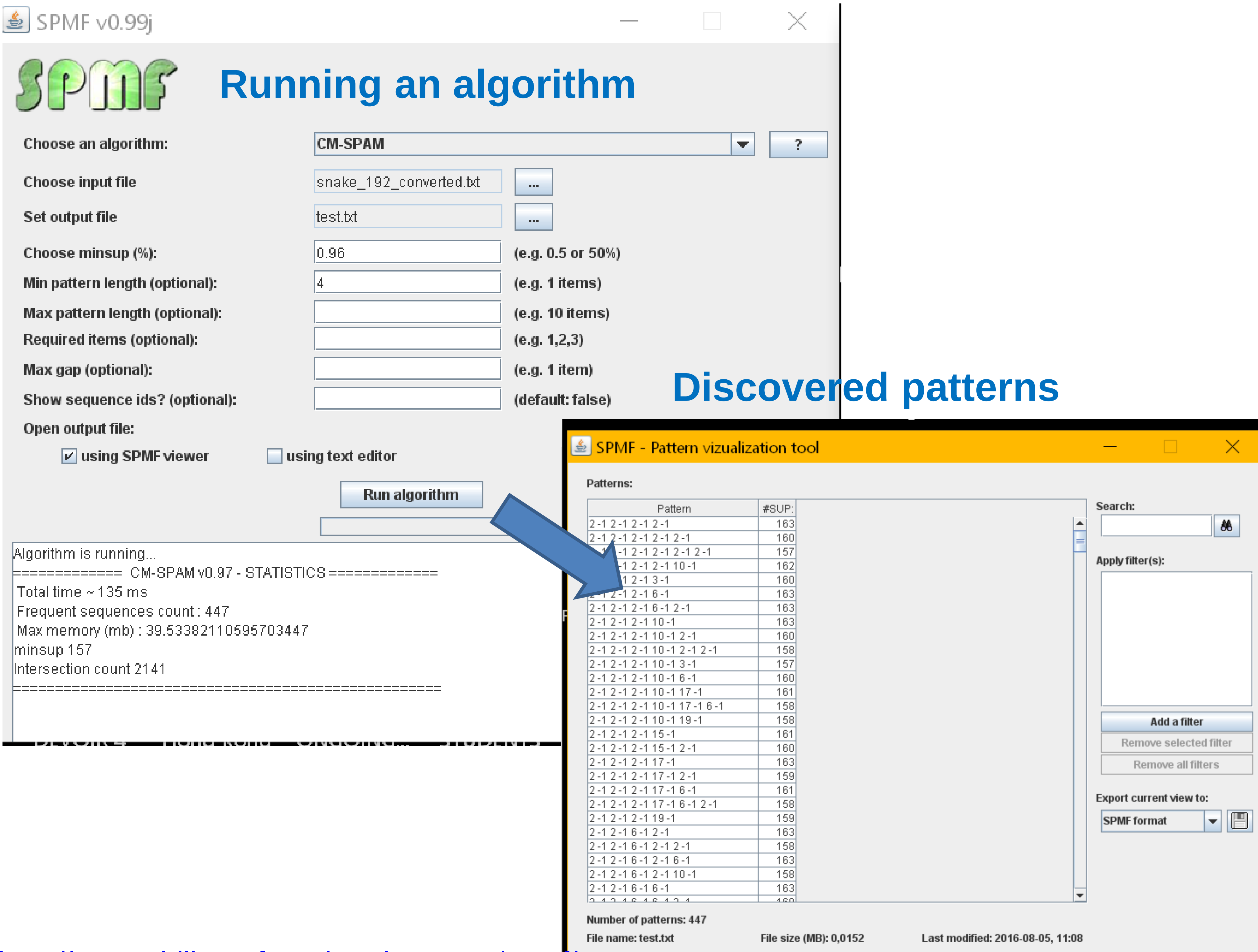
Moreover, SPMF can be used as a **standalone program** with a simple user interface or from the **command line**.

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




# Discovered patterns

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

# Some applications

**E-learning**

- Fournier-Viger, P., Faghihi, U., Nkambou, R., Mephu Nguifo, E.: CMRules: Mining Sequential Rules Common to Several Sequences. Knowledge-based Systems, Elsevier, 25(1): 63-76 (2012)
- Toussaint, Ben-Manson, and Vanda Luengo. “Mining surgery phase-related sequential rules from vertebroplasty simulations traces.” Artificial Intelligence in Medicine. Springer International Publishing, 2015. 35-46.
- Faghihi, Usef, Philippe Fournier-Viger, and Roger Nkambou. “CELTS: A Cognitive Tutoring Agent with Human-Like Learning Capabilities and Emotions.” Intelligent and Adaptive Educational-Learning Systems. Springer Berlin Heidelberg, 2013. 339-365.

# Some applications

**Manufacturing simulation**

- Kamsu-Foguem, B., Rigal, F., Mauget, F.: Mining association rules for the quality improvement of the production process. Expert Systems and Applications 40(4), 1034-1045 (2012)

**Quality control**

- Bogon, T., Timm, I. J., Lattner, A. D., Paraskevopoulos, D., Jessen, U., Schmitz, M., Wenzel, S., Spieckermann, S.: Towards Assisted Input and Output Data Analysis in Manufacturing Simulation: The EDASIM Approach. In: Proc. 2012 Winter Simulation Conference, pp. 257–269 (2012)

# Some applications

**Web page prefetching**

- Fournier-Viger, P. Gueniche, T., Tseng, V.S.: Using Partially-Ordered Sequential Rules to Generate More Accurate Sequence Prediction. Proc. 8th International Conference
on Advanced Data Mining and Applications, pp. 431-442, Springer (2012)

**Anti-pattern detection in service based systems,**

- Nayrolles, M., Moha, N., Valtchev, P.: Improving SOA antipatterns detection in Service Based Systems by mining execution traces. In: Proc. 20th IEEE Working Conference on Reverse Engineering, pp. 321-330 (2013)

**Embedded systems**

- Leneve, O., Berges, M., Noh, H. Y.: Exploring Sequential and Association Rule Mining for Pattern-based Energy Demand Characterization. In: Proc. 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings. ACM, pp. 1–2 (2013)

# Some applications

**Alarm sequence analysis**

- Celebi, O.F., Zeydan, E., Ari, I., Ileri, O., Ergut, S.: Alarm Sequence Rule Mining
  Extended With A Time Confidence Parameter. In: Proc. 14th Industrial Conference
  on Data Mining (2014)
- Ileri, Omer, and Salih Ergüt. “Alarm Sequence Rule Mining Extended With A Time Confidence Parameter.” (2014).

**Recommendation**

- Jannach, Dietmar, and Simon Fischer. “Recommendation-based modeling support for data mining processes.” Proceedings of the 8th ACM Conference on Recommender systems. ACM, 2014.

# Some applications

**Restaurant recommendation**

- Han, M., Wang, Z., Yuan, J.: Mining Constraint Based Sequential Patterns and Rules on Restaurant Recommendation System. Journal of Computational Information Systems 9(10), 3901-3908 (2013)

**Customer behavior analysis**

- Noughabi, Elham Akhond Zadeh, Amir Albadvi, and Behrouz Homayoun Far. “How Can We Explore Patterns of Customer Segments’ Structural Changes? A Sequential Rule Mining Approach.” Information Reuse and Integration (IRI), 2015 IEEE International Conference on. IEEE, 2015.

# Extensions

**Extensions** :

- **TRuleGrowth**: mining rules with a window size constraint
- **TopSeqRules**: mining the top-k sequential rules.
- **TNS**: mining the top-k non redundant sequential rules
- …