Session 3A
14:00
FIT 1-315

Efficient Mining of Top-K Sequential Rules

Philippe Fournier-Viger[1]

Vincent Shin-Mu Tseng[2]

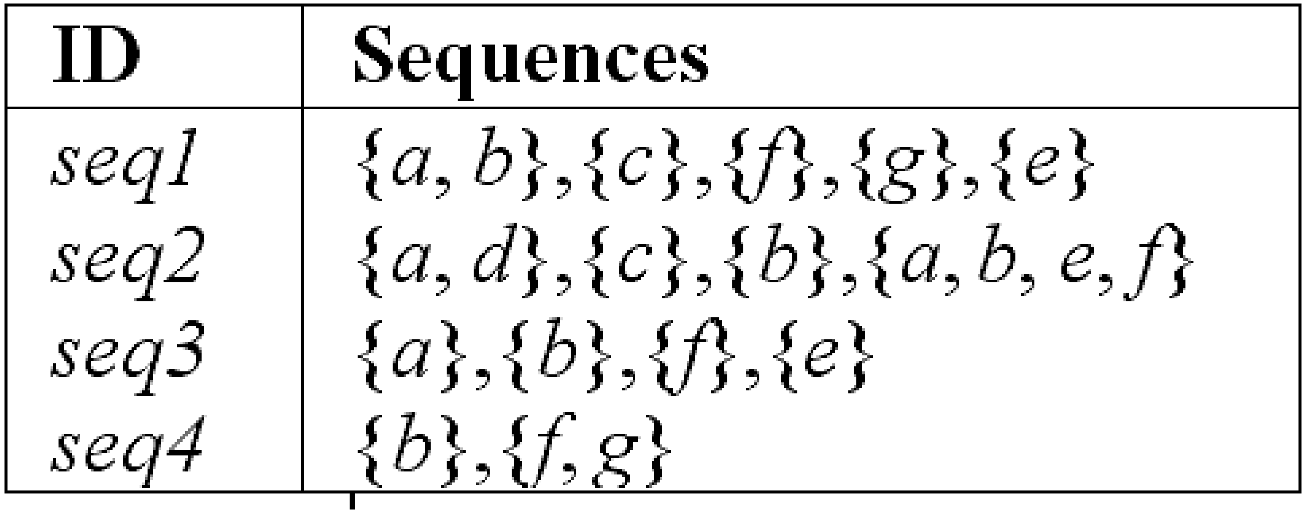
[1]University of Moncton, Canada

[2]National Cheng Kung University, Taiwan

18th December 2011 – **ADMA 2011** - Beijing

Sequence Database

- A set of sequences.
- Each **sequence** is an ordered list of transactions.
- A **transaction** is an unordered set of items (symbols), considered to occur simultaneously.

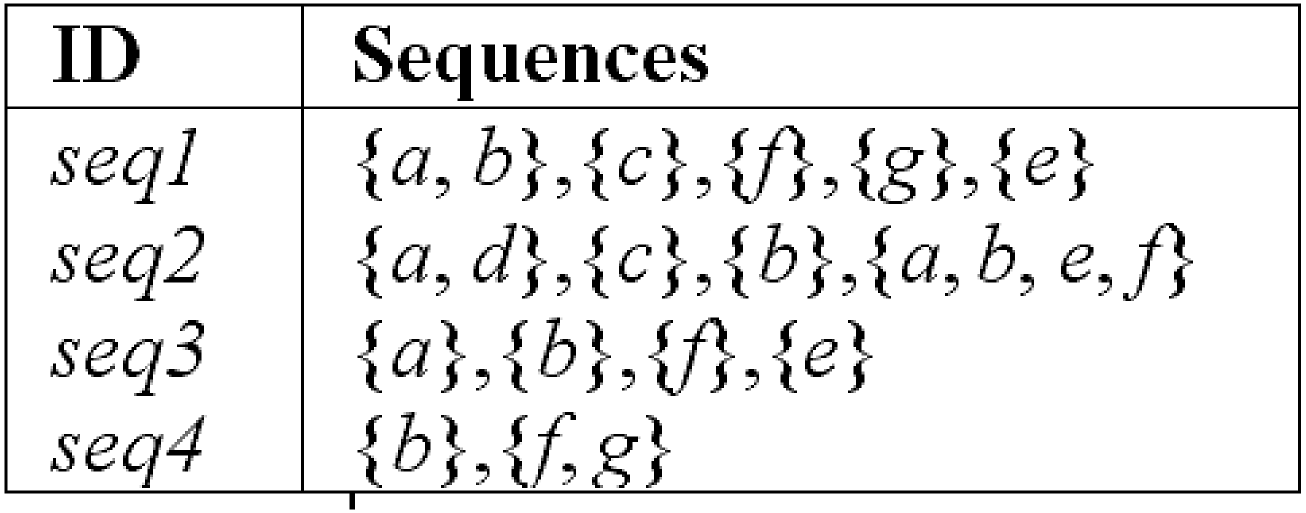


ID	**Sequences**
seq1	$\{a, b\},\{c\},\{f\},\{g\},\{e\}$
seq2	$\{a, d\},\{c\},\{b\},\{a, b, e, f\}$
seq3	$\{a\},\{b\},\{f\},\{e\}$
seq4	$\{b\},\{f, g\}$

- **Ex.:** click-stream data, market basket analysis, stock market data, bioinformatics, e-learning…

Sequential Pattern Mining

- **SPM** finds subsequences that are common to more than *minsup* sequences (Spade, PrefixSpan, GSP...).



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*}

- Ex.: {b}, {f}, {e} is a seq. pattern with support = 50 %.
- However, this pattern is misleading because {b}, {f} also appear 50 % of the time without {e} following.
- **To solve this problem,** we would need to consider the **confidence** of sequential patterns.

Sequential Rule Mining

- A **sequential rule** typically has the form X→Y and has a *confidence* and a *support.*
- **Algorithms for mining seq. rules** in a (1) single sequence, (2) across sequences or (3) **common to multiple sequences**.
- **Several applications:** stock market analysis (Das et al., 1998; Hsieh et al., 2006), weather observation (Hamilton & Karimi, 2005), drought management (Harms et al. 2002), alarm analysis, etc.

Mining sequential rule common to several sequences

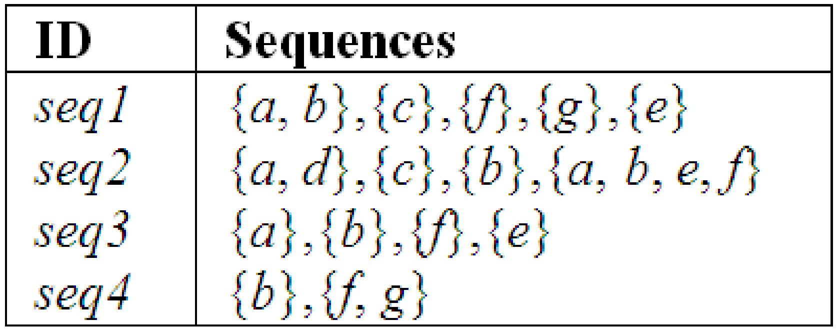
- RuleGen proposed by Zaki (2001).
- Rules of the form X → Y where X and Y are sequential patterns.
- Ex.: {a},{b},{c} → {d, e}
- **Problem**: rules can be too specific.
  - unlikely to match with a new sequence,
  - can have a low support value because they are very specific, despite that they may describe a common situation.
  - - ...

The definition used in this paper

- A **sequential rule** X⇒Y is a relationship between two disjoint and unordered itemsets X,Y.
- A sequential rule X⇒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 rules with a support and confidence not less than user-defined thresholds *minSup* and *minConf.*

Example

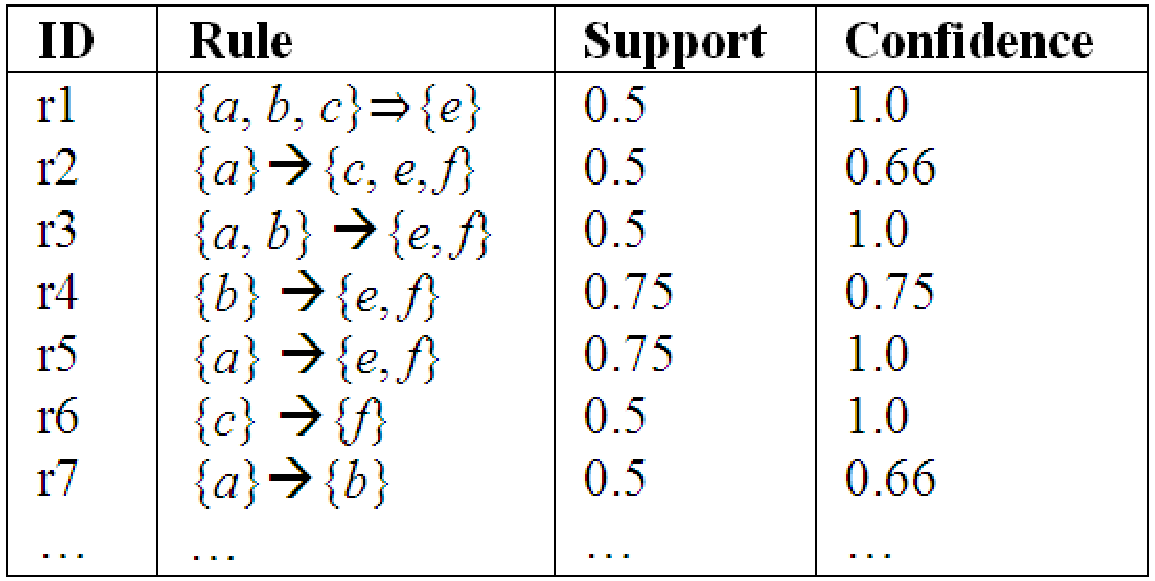
minSup= 0.5 and *minConf*= 0.5:



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	$\{a, b, c\} \Rightarrow \{e\}$	0.5	1.0
r2	$\{a\} \rightarrow \{c, e, f\}$	0.5	0.66
r3	$\{a, b\} \rightarrow \{e, f\}$	0.5	1.0
r4	$\{b\} \rightarrow \{e, f\}$	0.75	0.75
r5	$\{a\} \rightarrow \{e, f\}$	0.75	1.0
r6	$\{c\} \rightarrow \{f\}$	0.5	1.0
r7	$\{a\} \rightarrow \{b\}$	0.5	0.66
…	…	…	…

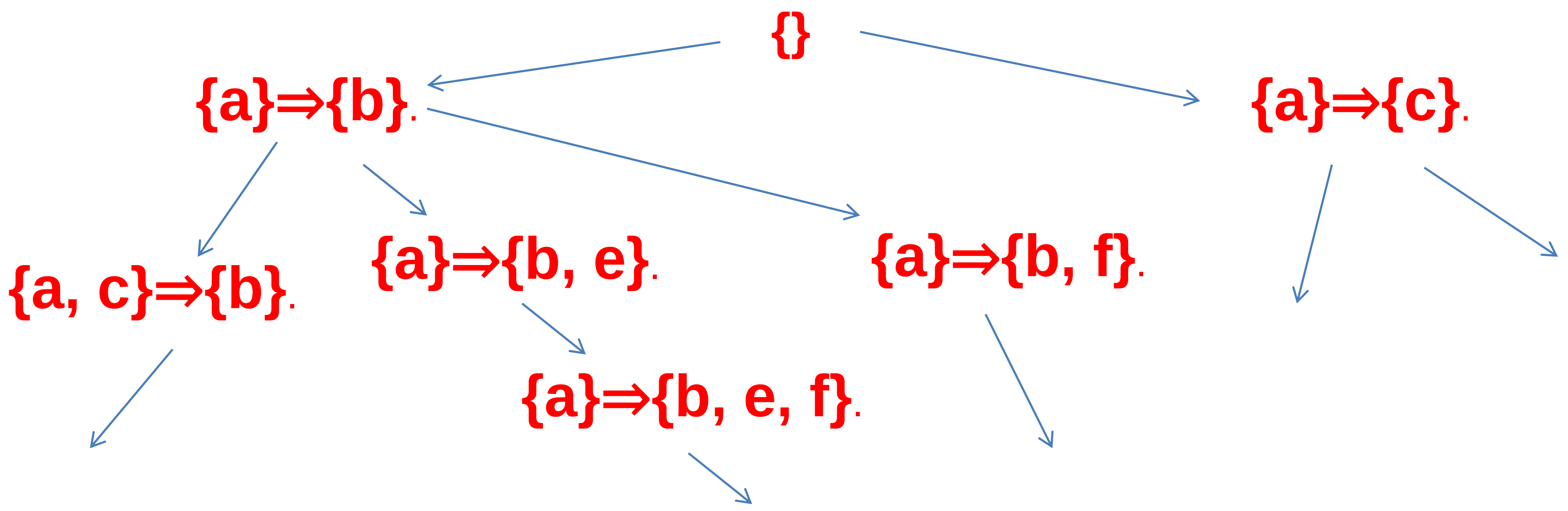
Some rules found

Current Algorithms

- **CMRules**: An association rule mining based algorithm for the discovery of sequential rules.
- **CMDeo**: An Apriori based algorithm for the discovery of sequential rules.
- **RuleGrowth**: A pattern-growth based algorithm. **(current best)**

RuleGrowth

Find larger rules by recursively scanning the database 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*).



The problem of setting *minsup*

- **Scenario:** A user wants to discover the top 1000 rules from a database and do not want to find more than 2000 rules.
- For *BMSWebView1*, the range of *minsup* values that will satisfy the user is 0.0011 to 0.0009.
- A user having no a priori knowledge of the database has only a 0.02 % chance of selecting a *minsup* value that will make him satisfied.
- Too high, not enough rules.
- Too low, performance deteriorates.

TopSeqRules

Main idea

- set *minsup* = 0.
- use the RuleGrowth strategy for rule generation.
- keep a set *L* that contains the current top-k rules found until now.
- when *k* rules are found, raise *minsup* to the lowest support of the rules in *L*.
- after that, for each rules added to *L*, raise the *minsup* threshold.

TopSeqRules (2)

- The resulting algorithm has poor execution time because the search space is too large.

$$3^d - 2^d + 1$$

- **Observation:** if we can find rules with higher support first, we can raise *minsup* more quickly and prune the search space.
- **How to define what is the most promising?** Our experiment show that the support is a good choice.
- We added a set R containing the *k* rules having the highest support.

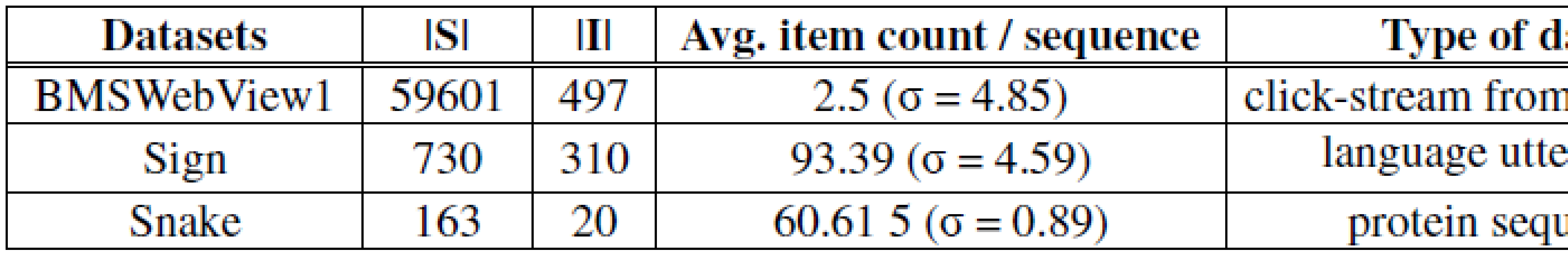
TopSeqRules (3) - optimizations

- We found that the choice of data structures for implementing L and R is also very important:
  - *L* : fibonnaci heap : O(1) amortized time for insertion and minimum, and O(log(n)) for deletion.
  - *R*: red-black tree: O(log(n)) worst case time complexity for insertion, deletion, min and max.
- Merging database scans

Experimental evaluation

- Three real-life datasets:

Table 1. Datasets characteristics

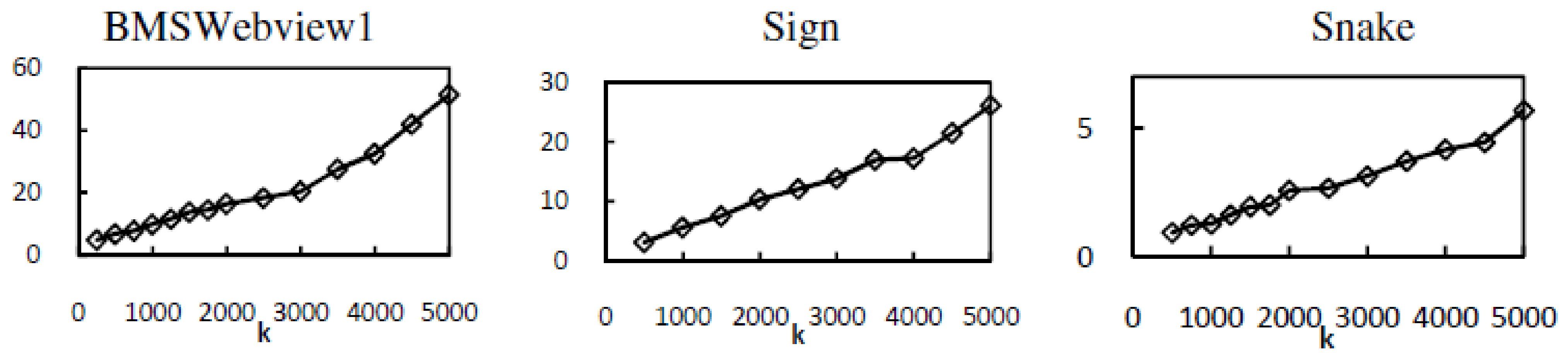


Datasets	|S|	|I|	Avg. item count / sequence	Type of d
BMSWebView1	59601	497	2.5 (σ = 4.85)	click-stream from
Sign	730	310	93.39 (σ = 4.59)	language utte
Snake	163	20	60.61 5 (σ = 0.89)	protein sequ

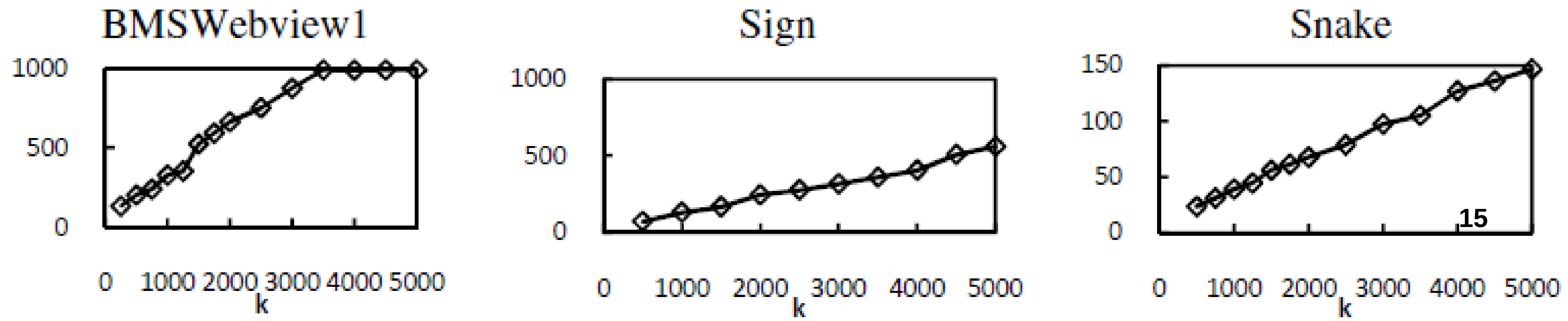
Results – influence of *k*

- Execution time an maximum memory usage grow linearly with *k*.

Execution time (seconds)



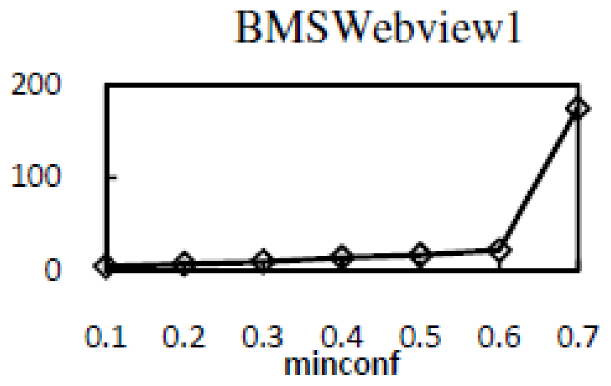
Maximum memory usage (megabytes)

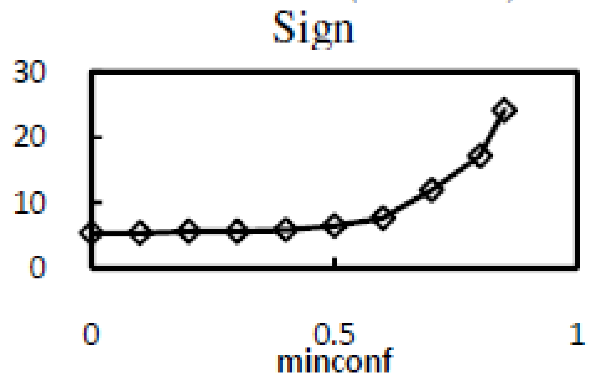


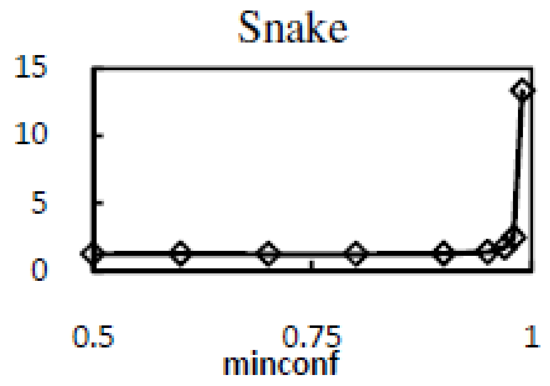
Results – influence of *minconf*

- Execution time and memory usage increase exponentially when *minconf* increases because more rules have to be generated.

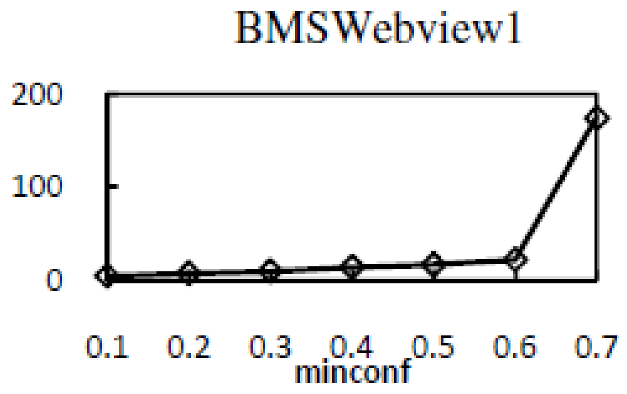
Execution time (seconds)

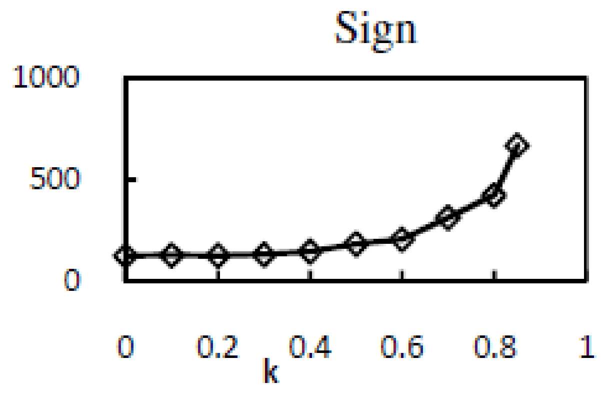


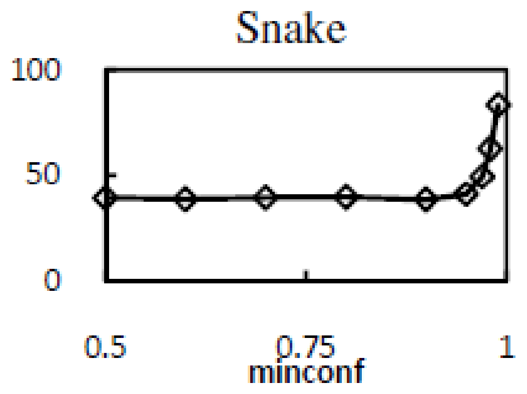




Maximum memory usage (megabytes)

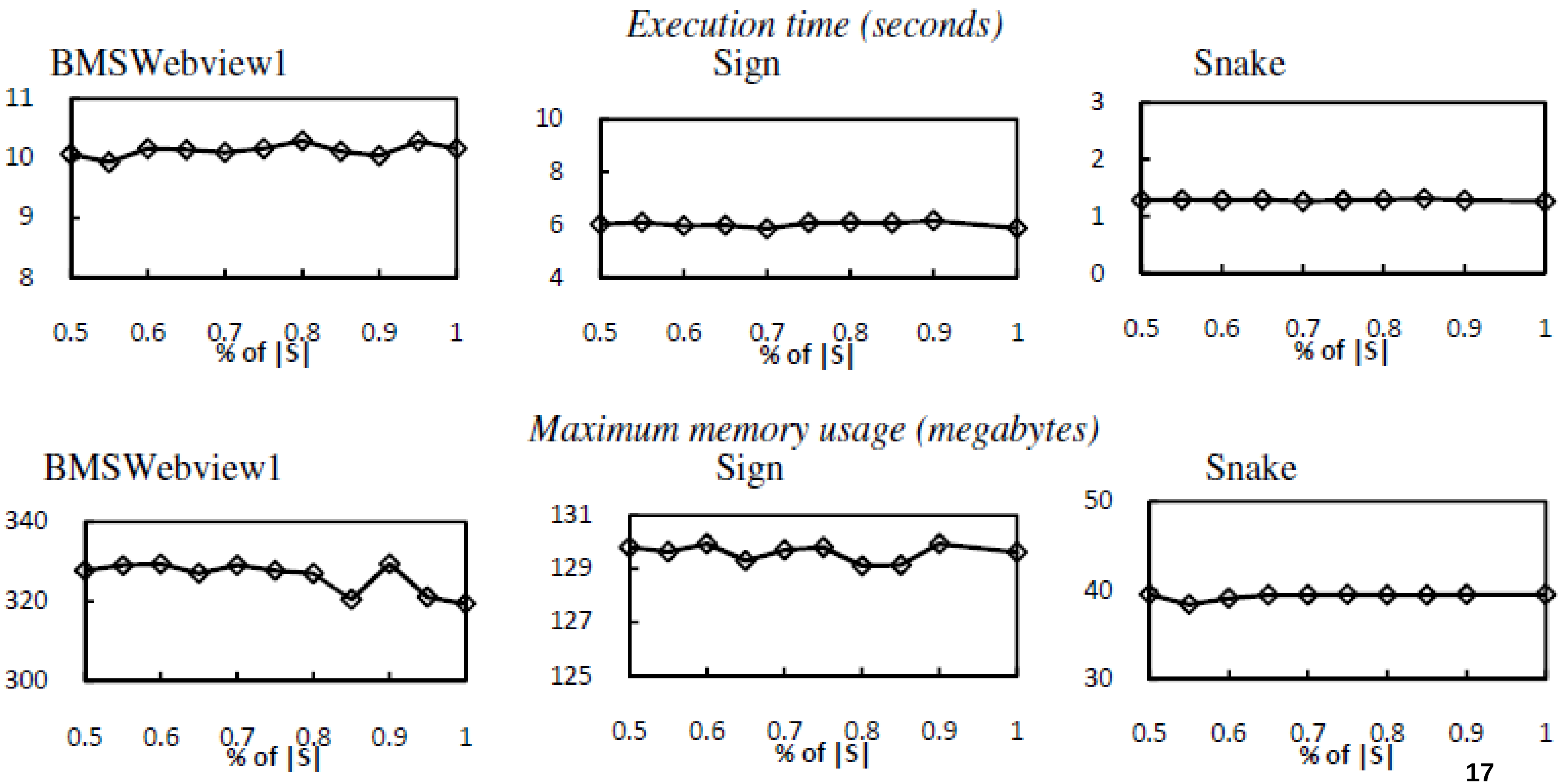






Results – influence of database size

- Execution time and memory increases slowly if the number of rules stay more or less the same.



Performance comparison

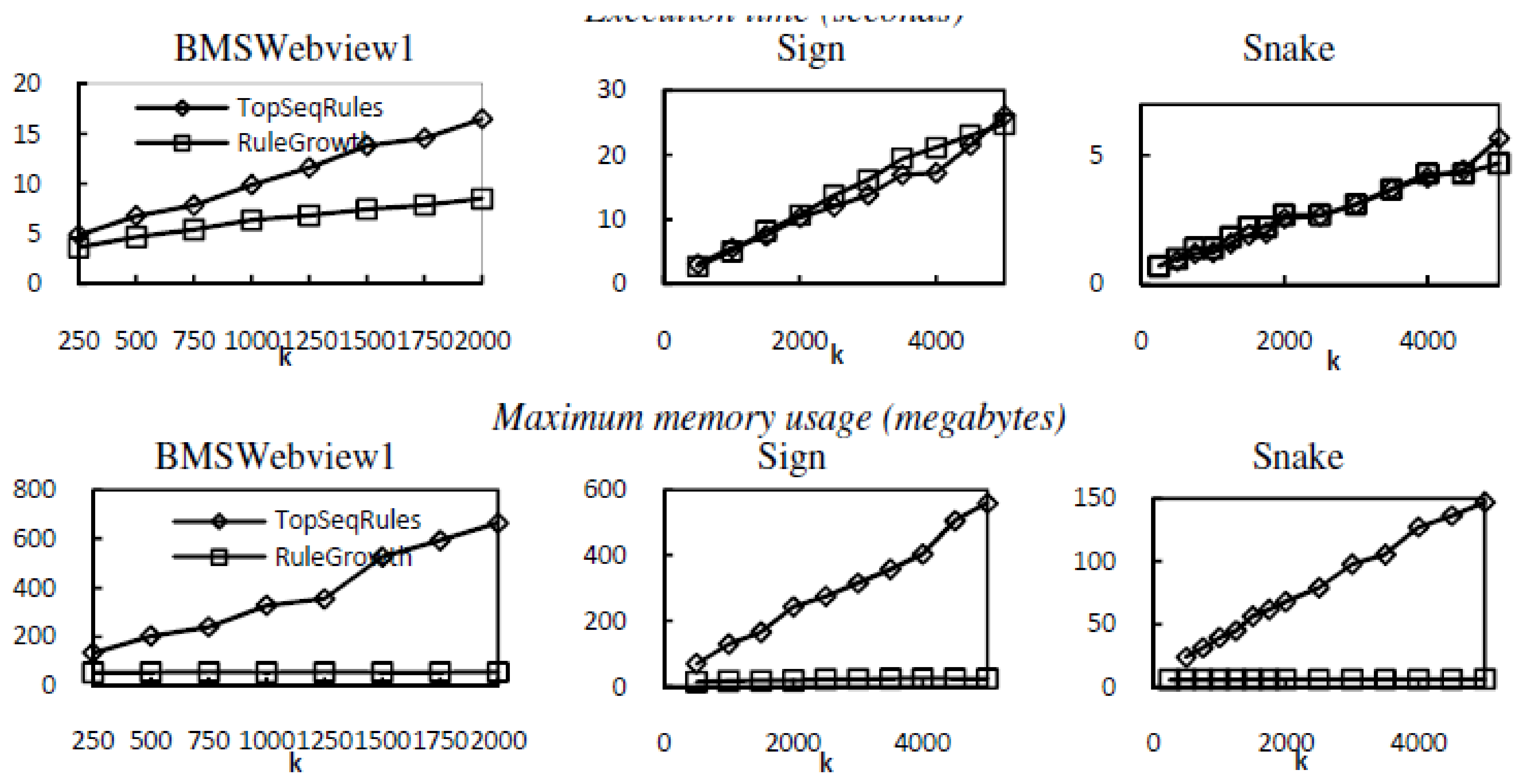
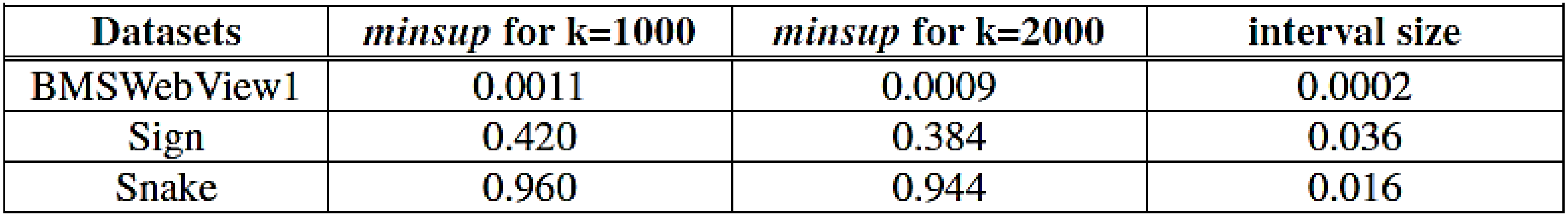


Fig. 9. Performance comparison for optimal parameters selection

Performance comparison (2)

Table 2. Interval of *minsup* values to find the top 1000 to 2000 rules for each dataset



Datasets	*minsup* for k=1000	*minsup* for k=2000	interval size
BMSWebView1	0.0011	0.0009	0.0002
Sign	0.420	0.384	0.036
Snake	0.960	0.944	0.016

- When *minsup* is chosen optimally, RuleGrowth has slightly better performance.
- However, setting *minsup* is very difficult.
- If *minsup* is set too low, RuleGrowth will not find any rule.
- If *minsup* is set too high, too many rules will be found and the performance deterioates

Conclusion

- We proposed an algorithm that let the user set *k,* the number of rules rules to be found.
- Excellent scalability: execution time linearly increases with *k*.
- the algorithm has no problem running in reasonable time and memory limits for *k* values of up to 5000 for all datasets.

Thank you. Questions?

Thanks to the organizers of ADMA 2011!

SPMF

Open source Java data mining software, 43 algorithms
http://www.phillippe-fournier-viger.com/spmf/

Thanks to the FQRNT funding programs.

Fonds de recherche sur la nature et les technologies Québec