Mining Top-K Non-Redundant Association Rules

Philippe Fournier-Viger[1]

Vincent Shin-Mu Tseng[2]

[1]University of Moncton, Canada

[2]National Cheng Kung University, Taiwan

UNIVERSITÉ DE MONCTON

11/12/2012

ISMIS 2012

Introduction

- **Association rule mining**
  - a **fundamental data mining task** with **wide applications.**
  - to find interesting associations between items in a **transaction database**.
- A **transaction database**

ID	Transactions
t_1	{a, b, c, e, f, g}
t_2	{a, b, c, d, e, f}
t_3	{a, b, e, f}
t_4	{b, f, g}

Association Rules

- **An association rule**
  - a relationship between two itemsets. e.g. {a, b} → {c}
  - **support :** the percentage of transactions of a database where the rule occurs.
  - **confidence :** the support of the rule divided by the support of its antecedent.
- **For example**:

ID	Transactions
t_1	{a, b, c, e, f, g}
t_2	{a, b, c, d, e, f}
t_3	{a, b, e, f}
t_4	{b, f, g}

{a, b} → {c} support = 0.5 confidence = 0.66

Association Rule Mining

- **Given** user-defined thresholds *minsup* and *minconf,* **discover** all rules with a support and confidence respectively higher or equal to these thresholds.

Problem #1

- **How to select the *minsup* and *minconf* thresholds ?**
  - too high, too few results
  - too low, too many results, performance often exponentially degrades
- **In real-life:**
  - time/storage limitation,
  - the user cannot analyze too many rules,
  - fine tuning parameters is time-consuming,

Example

- Consider the **Connect** dataset.
- To discover 1000 to 2000 rules.
- The range of *minsup* values that will satisfy him is 0.5060 to 0.5052.
- Thus, a user without a priori knowledge of the database has only a 0.08 % chance of selecting a *minsup* value that will make him satisfied.

Problem #2

- Many rules that are returned to the user are **redundant**.
- For example: {a} → {b, c} and {a} →{c} and other similar rules may have exactly the same support and confidence.
- Because of redundancy, it is more time-consuming to analyse the results.

Solutions?

- **For threshold selection:**
  - to discover the **top-k association rules (with a confidence higher than *minconf*).**
- **For the problem of redundancy:**
  - to discover **non-redundant sets of association rules**.
- **Could we address these two problems at the same time?**

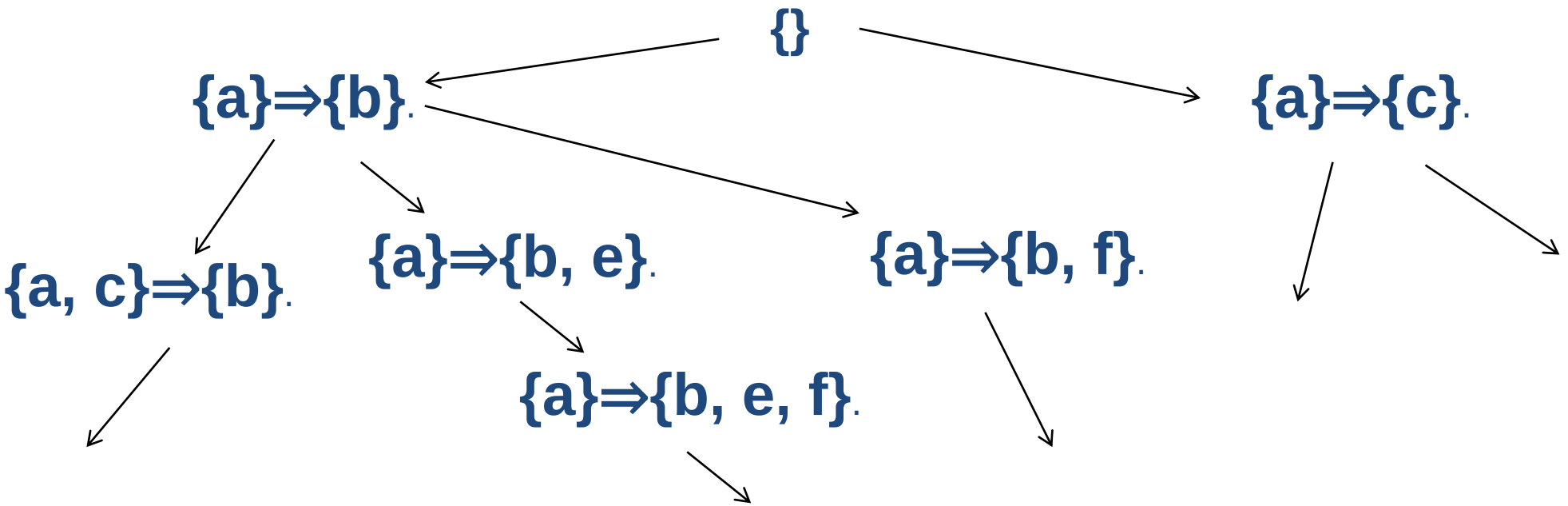
Problem statement

- **Problem**: Discovering **top-k non-redundant association rules.**
- An **important problem**
  - because from 12 % to 83 % of top-*k* rules are redundant for real datasets
- **Challenge:** We cannot eliminate redundancy by post-processing because less than *k* rules would be found.

The TNR Algorithm

Based on the **TopKRules** algorithm.

- depth-first search, search most promising rules first
- generate association rules directly by "rule-expansions"
- maintain a set **L** of current top-k rules during search
- several optimizations



The TNR Algorithm (cont'd)

- Def. of redundancy:
  - A rule X → Y is **redundant with respect to another rule** X1 → Y1 if it has the same support and confidence and $X1 \subseteq X$ and $Y \subseteq Y1$.
- Two strategies to eliminate redundancy:
  - **S1**: do not add a rule to L if it is redundant with respect to a rule already in L.
  - **S2**: if a rule in L is redundant with respect to a new rule that is found, then remove it from L.

The TNR Algorithm (cont'd)

The algorithm properties

- It is guaranteed to return non-redundant rules.
- It is not guaranteed to always generate the **top-k** non-redundant rules (because of strategy S2).

The TNR Algorithm (cont'd)

To increase the likelihood of obtaining an exact result, we can increase the size of L by a parameter Δ.

- If no more than Δ redundant rules are at the same time in L, the result is exact.
- Too costly to verify.

Performance comparison with TopKRules

Datasets	Runtime (s)		Maximum Memory Usage (MB)	
	TNR	TopKRules	TNR	TopKRules
Chess	8	1.49	269	72.12
Connect	283	25.51	699	403
Mushrooms	105	3.46	684	255
Pumsb	125	46.39	576	535

- For *k= 2000, minconf = 0.8*
- TNS is significantly more costly
  - because it has to perform additional checks
  - because it has to explore a larger search space

Influence of database size

For *k = 2000, minconf = 0.8*

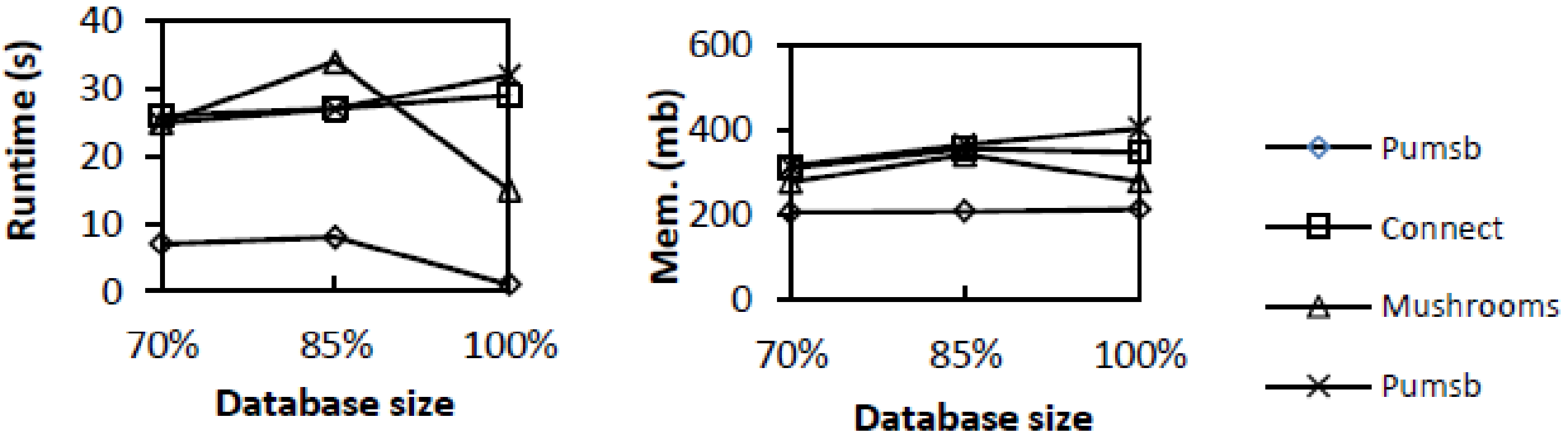


Fig. 2. Influence of the number of transactions

TNS has good scalability with respect to the number of transactions.

How many rules are discarded by each strategy?

Table 2. Rules discarded by each strategy for *minconf = 0.8, k=2000* and $\Delta = 0$

Dataset	# rules discarded by Strategy 1	# rules discarded by Strategy 2
Chess	961	10454
Connect	2732	15275
Mushrooms	39848	38627
Pumsb	803	3629

Higher for dense datasets (e.g. Mushrooms) than for sparse datasets (e.g. Pumsb)

What if we use the Δ parameter?

- *k = 2000, minconf* = 0.8.
- *Pumsb* dataset
- **Δ** = 125.
  - Runtime: 125 s.
  - Mem. Usage: 0.57 GB.
  - Rules discarded by S1: 803
  - Rules discarded by S2: 3,629
- **Δ** = 4000.
  - Runtime: 501 s.
  - Mem. Usage: 1.3 GB.
  - Rule discarded by S1: 3,454
  - Rule discarded by S2: 16,066

Performance comparison with FPGrowth

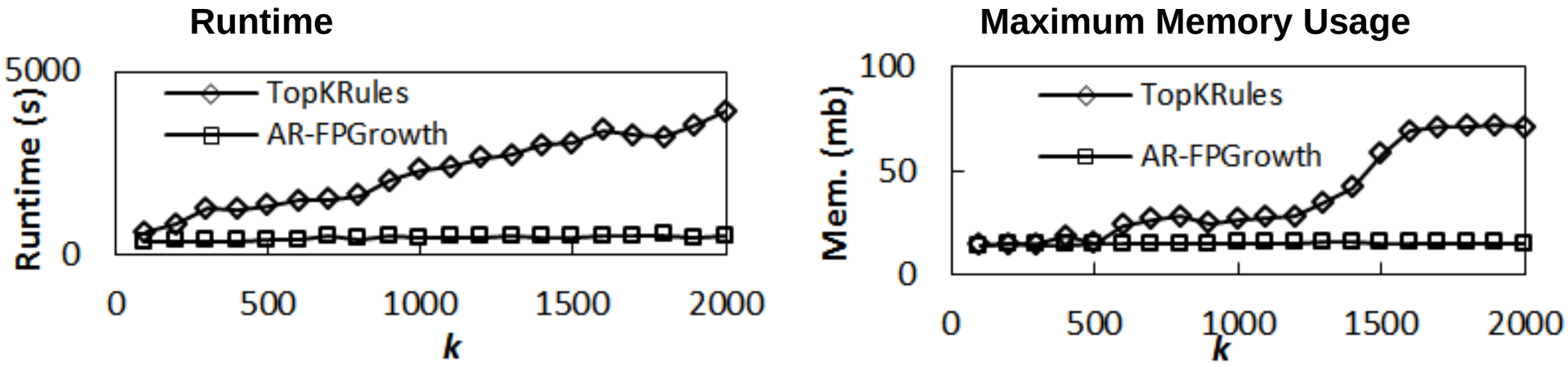


Fig. 9. Performance comparison for optimal *minsup* values for *chess*.

Table 4: 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
Chess	0.9415	0.9324	0.0091
Connect	0.5060	0.5052	0.0008
T25I10D10K	0.0120	0.0100	0.0020
Mushrooms	0.4610	0.4454	0.0156
Pumsb	0.6639	0.6594	0.0044

Results – influence of *k*

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

Table 2: Results for *minconf* = 0.8 and *k* = 100, 1000 and 2000

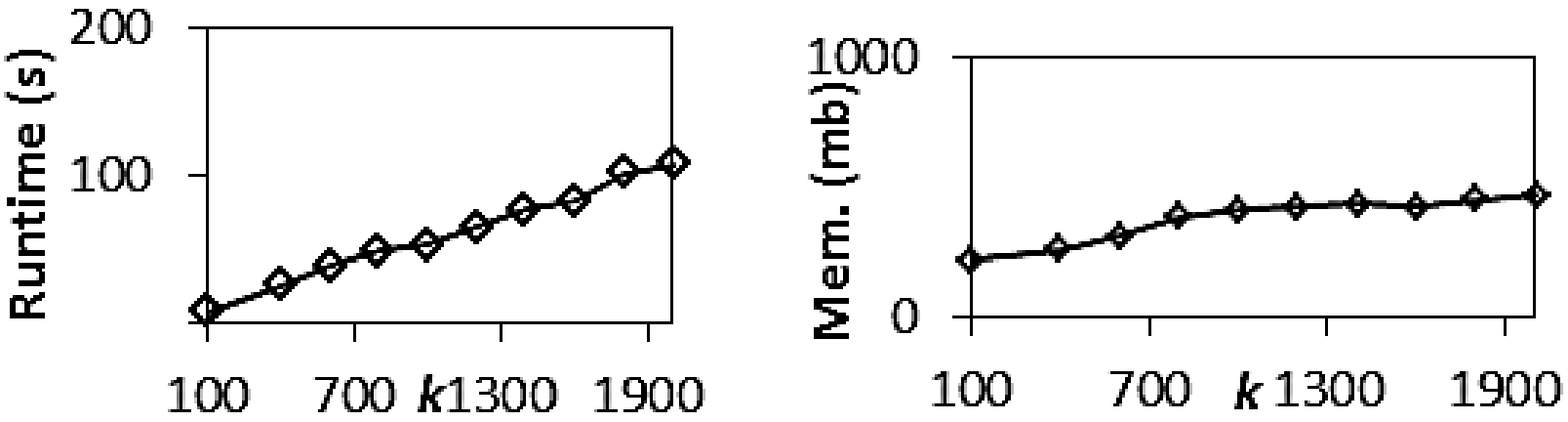


Fig. 6. Detailed results of varying *k* for the *Pumsb* dataset

Conclusion

- Two important problems in association rule mining are threshold selection and redundancy.
- These two problems have been addressed separately.
- We proposed **TNR,** an approximate algorithm to mine the **top-k non redundant association rules**.
- More costly than **TopKRules**. But it eliminates a great deal of redundancy.
- All source code available on SPMF website under GPL 3 license.

Open source Java data mining software, 50 algorithms
http://www.phillippe-fournier-viger.com/spmf/