ASE BigData 2015

7 - 9 October 2015

Kaohsiung, Taiwan

Efficient Incremental High Utility Itemset Mining

Philippe Fournier-Viger[1]

Jerry Chun-Wei Lin[2]

Ted Gueniche[1]

Prashant Barhate[3]

[1]University of Moncton, Canada

[2]Harbin Inst. of Techn., Shenzhen Graduate School

[3]Infosys, Ltd.

High Utility Itemset Mining

Input: transaction database with quantities

TID	items
T_0	a(1), b(5), c(1), d(3), (e,1)
T_1	b(4), c(3), d(3), e(1)
T_2	a(1), c(1), d(1)
T_3	a(2), c(6), e(2)
T_4	b(2), c(2), e(1)

unit profit table

item	unit profit
a	5 $
b	2 $
c	1 $
d	2 $
e	3 $

a threshold ***minutil***

Output: high-utility itemsets (HUIs),
i.e. itemsets having a **utility** $\geq$ ***minutil***

A full example

Transation database

TID	items
T_0	a(1), b(5), c(1), d(3), (e,1)
T_1	b(4), c(3), d(3), e(1)
T_2	a(1), c(1), d(1)
T_3	a(2), c(6), e(2)
T_4	b(2), c(2), e(1)

Unit profit table

item	unit profit
a	5 $
b	2 $
c	1 $
d	2 $
e	3 $

Suppose that
***minutil* = 25 $** →

High utility itemsets

{a,c} : 28$
{a,b,c,d,e}: 25 $
{b,c,d}: 34 $
{b,c,e} : 37 $
{b,d,e} : 36 $
{c, e}: 27$
{a,c,e}: 31 $
{b,c} : 28 $
{b,c,d,e}: 40 $
{b,d} : 30 $
{b,e} : 31 $

Problem

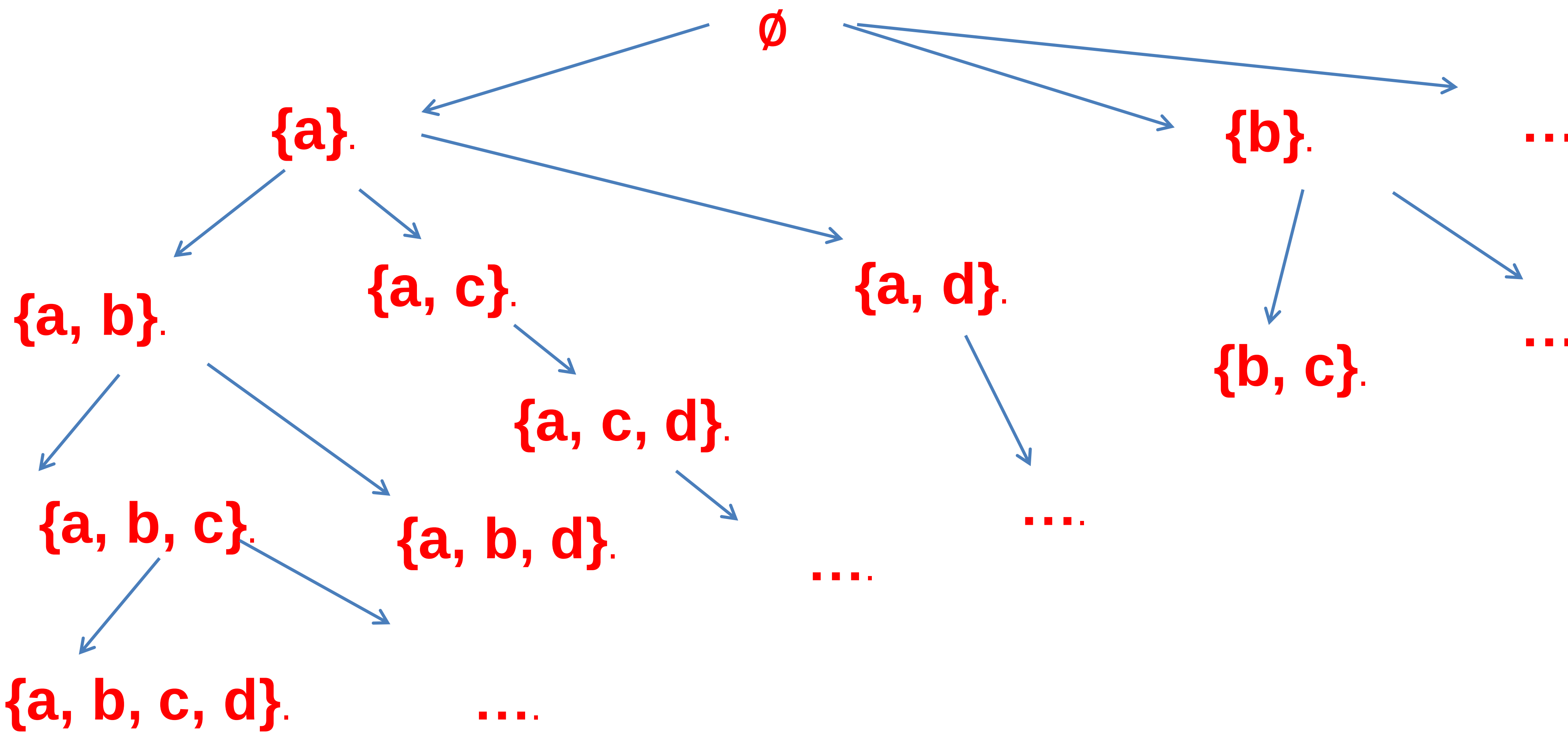
- Most algorithms assume that the database is static.
- A few incremental algorithms: **HUI-LIST-INS**, etc.
- **Contribution**: a faster algorithm (**EIHI**) to update high-utility itemsets when new transactions are inserted.

	TID	items
D	T_0	a(1), b(5), c(1), d(3), (e,1)
	T_1	b(4), c(3), d(3), e(1)
	T_2	a(1), c(1), d(1)
	T_3	a(2), c(6), e(2)
	T_4	b(2), c(2), e(1)
N	**T_5**	**b(2), c(2), e(1)**

EIHI

Extends the FHM algorithm

– Find larger itemsets with depth-first search by appending items one at a time.



EIHI

– Create a vertical structure named **Utility-List** for **each item.**

Utility list of {a}

TID	util	rutil
T_1	5	3
T_2	10	17
T_3	5	25

u({a}) = 20

\+ join

Utility list of {e}

TID	util	rutil
T_2	6	5
T_3	3	5
T_4	3	0

u({e}) = 12

→

Utility list of {a, e}

TID	util	rutil
T_2	16	5
T_3	8	5

u({a,e}) = 24

– The exact utility of an itemset is obtained by joining utility-lists of smaller itemsets (no need to scan database).

– **Pruning** using remaining utility in utility lists

EIHI

– Create a vertical structure named **Utility-List** for **each item.**

Utility list of {a}

TID	util	rutil
T_1	5	3
T_2	10	17
T_3	5	25

u({a}) = 20

+ join

Utility list of {e}

TID	util	rutil
T_2	6	5
T_3	3	5
T_4	3	0

u({e}) = 12

→

Utility list of {a, e}

TID	util	rutil
T_2	16	5
T_3	8	5

u({a,e}) = 24

– The exact utility of an itemset is obtained by joining utility-lists of smaller itemsets (no need to scan database).

– **Pruning** using remaining utility in utility lists

EIHI - first run

Assume that EIHI is run for the first time on a database D

When a high utility itemset is found

- It is inserted in a trie structure with its utility,

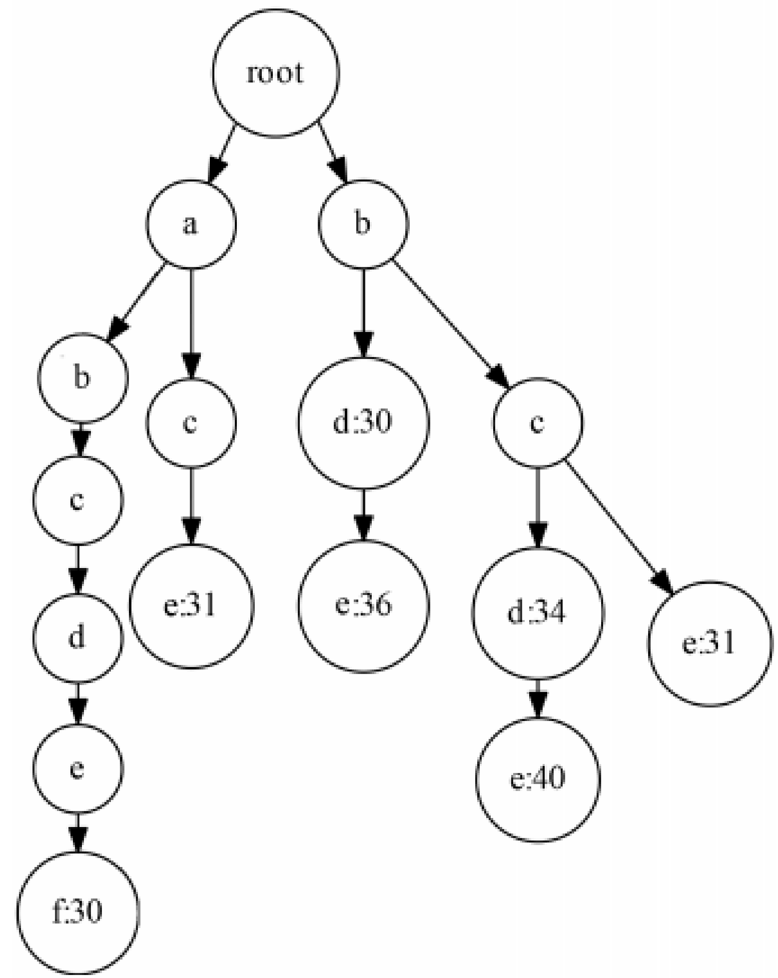


Figure 1: The HUI-trie structure

EIHI – second run

- Now assume that the database **D** is updated with some new transactions **N**.
- EIHI is run again by considering only items in **N**.
- Each utility-list store the transactions from **N** separately from those in **D**.

Utility list of {e}

	TID	util	rutil
D	T_2	6	5
	T_3	3	5
	T_4	3	0
N	T_5	3	0

EIHI – second run

When a high utility itemset is found

- If the itemset is already in the trie, its utility is updated.
- Otherwise, it is inserted in the trie

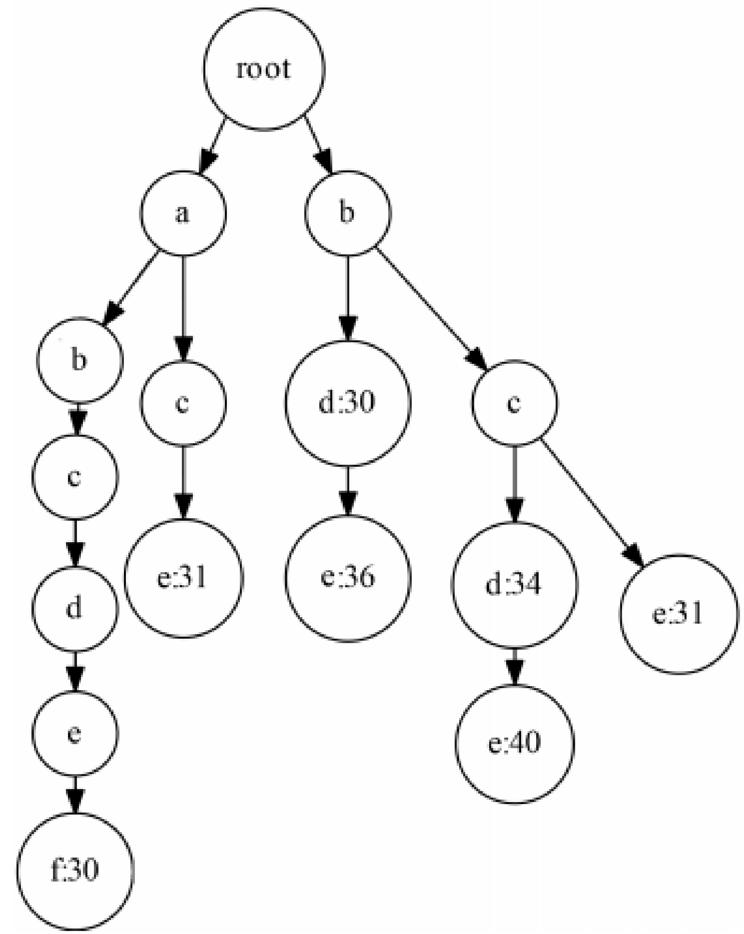


Figure 1: The HUI-trie structure

Property: A HUI in **D** will remain a HUI in **D+N.**

EIHI – second run

- **Pruning property 1**: If an itemset **X** does not appear in **N**, the itemset **X** and its extensions do not need to be considered in the current run.

- This is because in that case the utility of **X** in **D+N** will be the same as in **D**.

EIHI – second run

- **Pruning property 2**: For an itemset **X**, if its utility + remaining utility in **D** plus the utility + remaining utility in **N** is less than minUtil, extensions of **X** are not explored.
- This is a generalization of the pruning criteria used by FHM.

Utility list of {e}

	TID	util	rutil
D	T_2	6	5
D	T_3	3	5
D	T_4	3	0
N	T_5	3	0

= 25

EIHI – second run

- At the end of the run, transactions from **N** are added to transactions from **D** in each utility-list to prepare for the next run.

D

TID	util	rutil
T_2	6	5
T_3	3	5
T_4	3	0
T_5	3	0

N

→

D

TID	util	rutil
T_2	6	5
T_3	3	5
T_4	3	0
T_5	3	0

N …….

Some additional ideas are discussed in the paper!

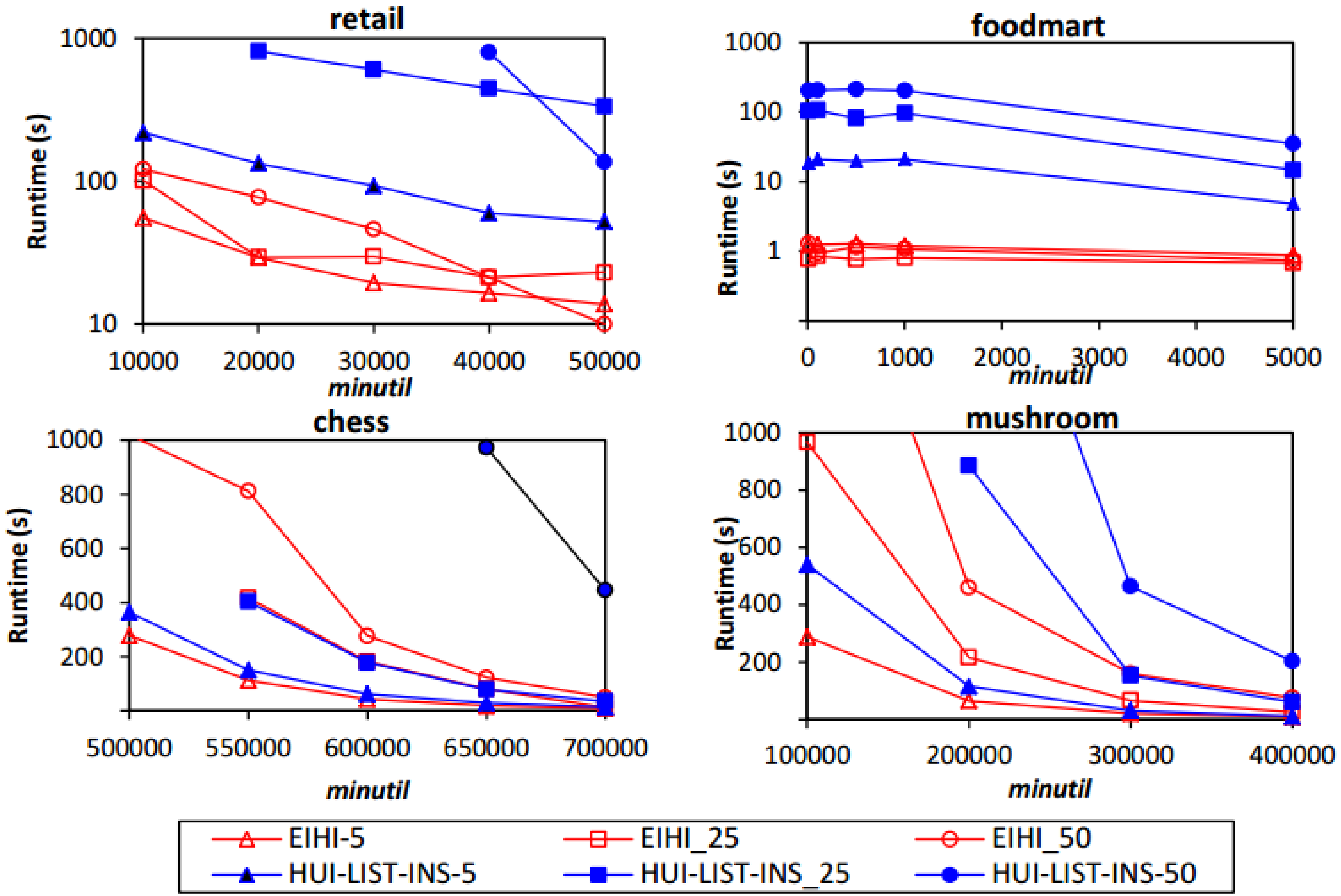
Experimental Evaluation

Four datasets

Dataset	transaction count	distinct item count	avg. transaction length
Retail	88,162	16,470	23
Foodmart	4,141	1,559	4.4
Chess	3,196	75	35
Mushroom	8,124	120	23

- **Foodmart:** real utility values
- **Other datasets:** Unit profit between 1 and 1000 and quantities between 1 and 5 (normal distribution)
- EIHI vs HUI-LIST-INS
- Java, Windows 7, 5 GB of RAM

Execution time



- EIHI is **up to 220 times faster than** HUI-LIST-INS.
- Larger gap for sparse datasets
- he gap increases when the number of update increases.
- EIHI has very similar memory usage to HUI-LIST-INS.

Conclusion

- We presented a new incremental high-utility itemset mining algorithm named **EIHI.**
- Introduced several novel ideas.
- **Results**
  - up to **220 times faster than the state-of-the-art algorithm**
  - similar memory usage

- **Perspectives**:
  - Further optimizations
  - Extension for on-shelf high utility itemset mining, closed high utility itemset mining and top-k high utility itemset mining.

SPMF **Open source Java data mining software**, > 80 algorithms
http://www.phillippe-fournier-viger.com/spmf/

Thank you. Questions?

Open source Java data mining software, >80 algorithms
http://www.phillippe-fournier-viger.com/spmf/

Acknowledgement. This research was partially supported by the Natural Scientific Research Innovation Foundation in Harbin Institute of Technology under grant HIT.NSRIF.2014100, by the Tencent Project under grant CCF-TencentRAGR20140114, and by the National Natural Science Foundation of China (NSFC) under grant No.61503092.

A few references

1. P. Fournier-Viger, C.-W. Wu, S. Zida and V. S. Tseng. FHM: Faster high-utility itemset mining using estimated utility co-occurrence pruning. In Proc. 21st Intern. Symp. on Methodologies for Intell. Syst., pages 83–9. 2014.
2. J. C.-W. Lin, T. P. Hong, G. C. Lan, J. W. Wong and W. Y. Lin. Incrementally Mining High Utility Patterns based on Pre-large Concept. Applied Intelligence, 40:343–347, 2014.
3. J. C.-W. Lin, W. Gan, T.P. Hong and J. S. Pan. Incrementally Updating High-Utility Itemsets with Transaction Insertion. In Proc. 10th Intern. Conf. on Advanced Data Mining and Appl., pages 44–56. 2014.
4. P. Fournier-Viger and S. Zida. FOSHU: Faster On-Shelf High Utility Itemset MiningˆaA¸S with or without ˘ negative unit profit. In Proc. 30th Symposium on Applied Computing, pages 857–864. 2015. [6]
5. P. Fournier-Viger, C. W. Wu and V. S. Tseng. Novel Concise Representations of High Utility Itemsets using Generator Patterns. In 10th Intern. Conf. on Advanced Data Mining and Applications, pages 30–43. 2014.