PAPER

Appendix: A transaction transposition for QB

Siriwat KASAMWATTANAROTE[†,††a)], ***Nonmember***, **Yusuke UCHIDA**[†††*b)], ***Member***, ***and*** **Shin'ichi SATOH**[††*c)], ***Senior Member***

SUMMARY Once we do mining on a top-k relevant images with a very large vocabulary size, we found the patterns may response to several duplicate objects on different images. This leads to a time consuming problem on our mining step. Therefore, a technique called transposition of transaction helps reducing a mining space, which then map the mining result back to the original aspect by using Galois connection through an inverse relationship on a complete lattice. By doing a this, we can save a lot of mining time on any FIM algorithms.

key words: *Frequent itemset mining, Visual word mining, Query bootstrapping, Transaction transposition, Galois connection.*

1. Transaction Transposition for QB

Continue from our paper "Query Bootstrapping", before doing a transaction transposition, we need to check whether our case satisfying the condition said in Galois mapping as follow:

> "The mapping f is antitone and there exists an antitone mapping g from P to P' such that the composition mapping are extensive. – (GM)[1]"

Let P and P' are two complete lattices generated from a transaction database T and a transposed transaction T^T. Using our toy example on a table 1, we then found total patterns of P and P' are isomorphic to each other on the lattice as shown in a Fig. 2.

As we found out our target patterns can be mined from T and will be faster with T^T, however, the meaning of both pattern results are different. Mining patterns from original transaction database means, *we are finding which visual word sets shared among images*, where $p \in P : p = \{i_1, i_2, i_3 ... i_m\}$. In contrast, mining patterns from a transposed transaction means, *we are finding which images contain similar visual words.*, where $p' \in P' : p' = \{t_1, t_2, t_3 ... t_{k'}\}$. In order to utilize a patterns P', we need a mapping function $f : P' \rightarrow P$ as follow:

$$f(P') = \forall p'_{j',t'} \in P' : \mathbf{A} \tag{1}$$

Manuscript received November 8, 2014.
Manuscript revised November 23, 2014.
[†]The Graduate University for Advanced Studies, Tokyo, Japan
[††]National Institute of Informatics, Tokyo, Japan
[†††]KDDI R&D Laboratories, Inc., Saitama, Japan
[*]University of Tokyo, Tokyo, Japan
a) E-mail: siriwat@nii.ac.jp
b) E-mail: ys-uchida@kddilabs.jp
c) E-mail: satoh@nii.ac.jp
DOI: 10.1587/transinf.E0.D.1

Img. I_k	Trans. t_k
I_1	$t_1 = \{i_1, i_2, i_4, i_6\}$
I_2	$t_2 = \{i_2, i_5, i_8\}$
I_3	$t_3 = \{i_2, i_3, i_9\}$
I_4	$t_4 = \{i_1, i_2, i_4, i_7\}$
I_5	$t_5 = \{i_2, i_3, i_8\}$

Pattern	*support*
$\{i_2\}$	60%
$\{i_3\}$	40%
$\{i_8\}$	40%
$\{i_1, i_4\}$	40%
$\{i_3, i_8\}$	20%
$\{i_1, i_4, i_7\}$	20%
$\{i_2, i_3, i_9\}$	20%
$\{i_2, i_5, i_8\}$	20%
$\{i_1, i_2, i_4, i_6\}$	20%

Table 1: (left) Input simple transactions of top 5 images. (right) Output corresponding patterns found with *minsup* value 10%

where

$$\mathbf{A} = \bigwedge \begin{bmatrix} [T(p'_{1,1}) \wedge T(p'_{1,2}) \wedge \ldots \wedge T(p'_{1,k'})]_1 \\ [T(p'_{2,1}) \wedge T(p'_{2,2}) \wedge \ldots \wedge T(p'_{2,k'})]_2 \\ \vdots \\ [T(p'_{j',1}) \wedge T(p'_{j',2}) \wedge \ldots \wedge T(p'_{j',k'})]_j \end{bmatrix} \tag{2}$$

and

$$\mathbf{A} = P \tag{3}$$

where P' will be map to P through a transaction T, $T(x)$ will return a set of items on original T, and the total number of patterns on both space will be the same, as $\|P'\| = \|P\|$ or $j' = j$.

To be more clear on what the function does is that, from the patterns P', we map back each item founded in p', which corresponds to a *transaction id* t', to an original transaction database T. The actual set of items i_m on each mapped transaction will be checked to find which item appear on all transactions t'. And such item i_m will be collected to build p as a mapped pattern from $p' \rightarrow p$. In the final sense, we will discover patterns several order of magnitudes faster than a traditional way (see a timing report on both $FIM(s)$ and $FIM^T(s)$ in our full-paper).

References

[1] O. Ore, "Galois connections," Trans. Amer. Math. Soc., 1944.
[2] "Lattice miner, http://sourceforge.net/projects/lattice-miner/."

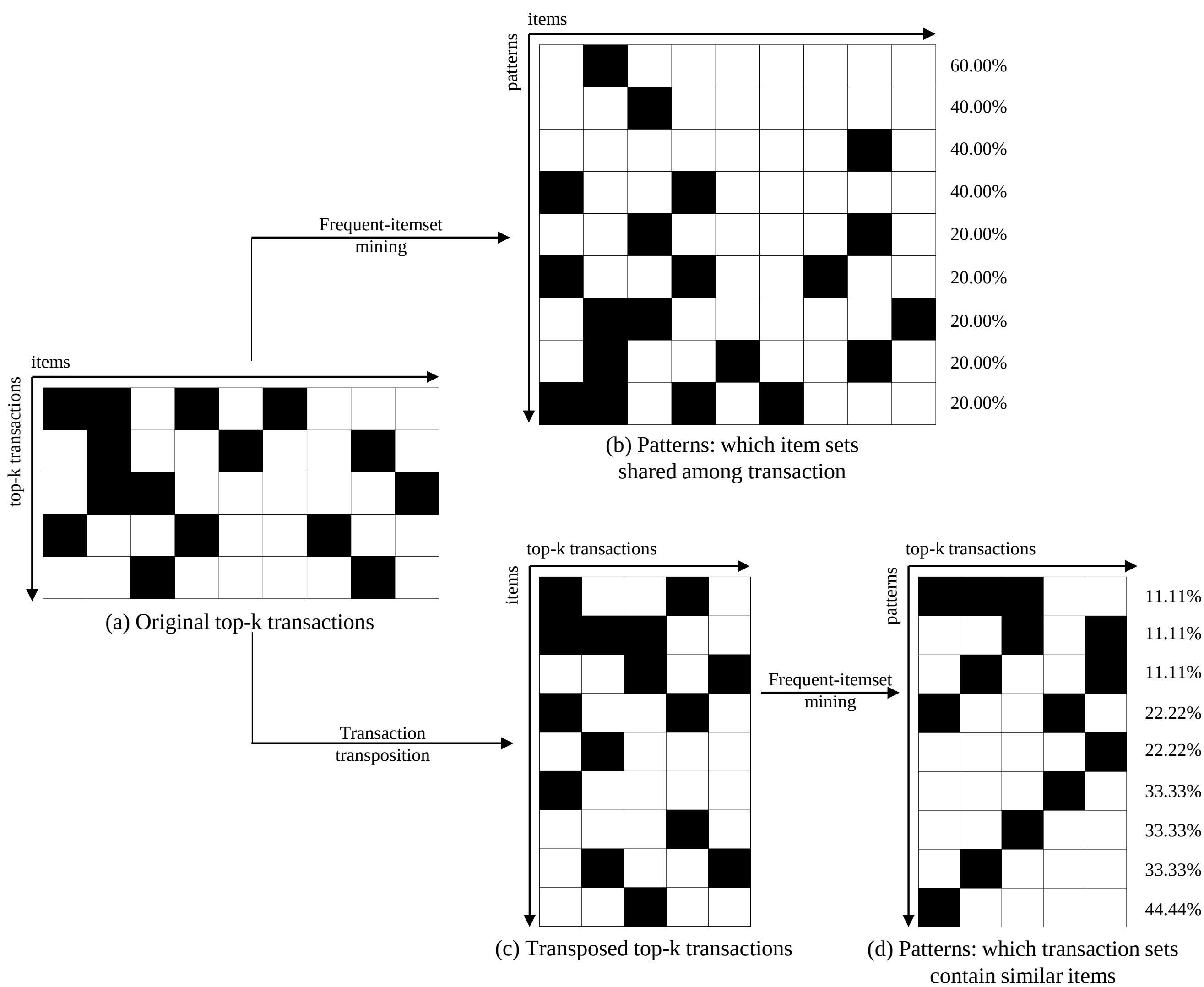

Fig. 1: (a) An original transaction database (T). (c) A transaction database after transposed (T^T) for speed-up FIM process (b),(d) Patterns (P and P') discovered from FIM using a normal database and a transposed database respectively.

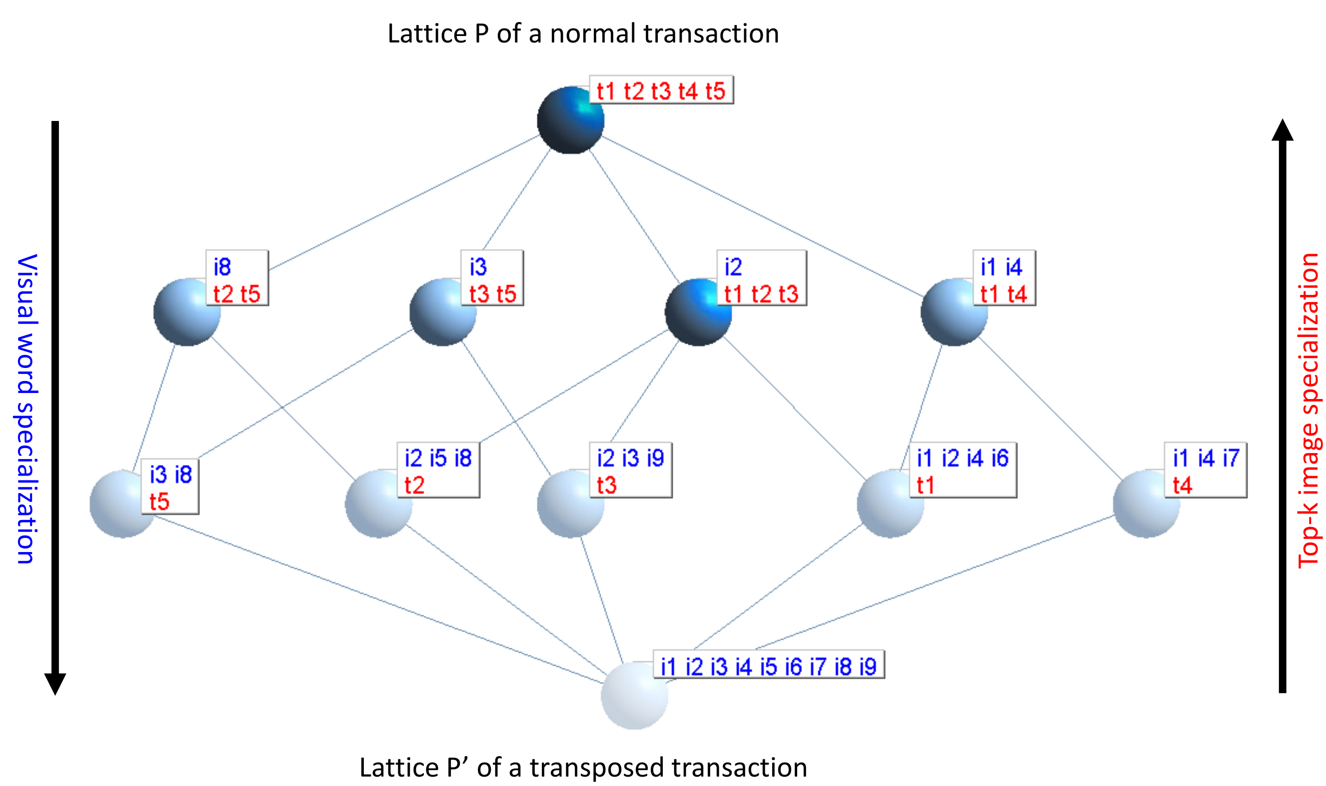

Fig. 2: Two complete lattices[a] of (top-down) a toy example transaction database and (bottom-up) a transposed transaction show an isomorphic property which satisfy the Galois mapping condition.

[a]The lattices of this toy example is visualized by Lattice Miner. [2]