CGT: a vertical miner for frequent equivalence classes of itemsets (extended abstract)

Laszlo Szathmary[1], Petko Valtchev[2], Amedeo Napoli[3], Marton Ispany[1], and Robert Godin[2]

[1] University of Debrecen, Faculty of Informatics, Department of IT, H-4010 Debrecen, Pf. 12, Hungary
{szathmary.laszlo, ispany.marton}@inf.unideb.hu
[2] Dépt. d'Informatique UQAM, C.P. 8888, Succ. Centre-Ville, Montréal H3C 3P8, Canada
{valtchev.petko, godin.robert}@uqam.ca
[3] LORIA (CNRS - Inria NGE - Université de Lorraine) BP 239, 54506 Vandœuvre-lès-Nancy Cedex, France
napoli@loria.fr

Abstract. In this extended abstract we present a vertical, depth-first algorithm that outputs frequent generators (FGs) and their associated frequent closed itemsets (FCIs). The proposed algorithm –called *CGT*– is a single-pass algorithm and it explores frequent equivalence classes in a dataset.

Introduction. In data mining, frequent itemsets (FIs) and association rules play an important role. Due to the high number of patterns, various concise representations of FIs have been proposed, of which the most well known representations are the FGs and the FCIs. There are a number of methods in the literature that target both FCIs and FGs, but most of these algorithms are levelwise methods. It is known that depth-first algorithms usually outperform their levelwise competitors. Here we present briefly a single-pass, depth-first, vertical FG+FCI miner.[1,2]

The CGT algorithm. *CGT* is a vertical itemset mining algorithm for finding frequent equivalence classes. *CGT* is based on *Talky* [1], where *Talky* is a modified version of *Eclat* [2]. *Eclat* and *Talky* produce the same output, i.e. they find all FIs in a dataset. However, *Talky* uses a different traversal called reverse pre-order strategy. This traversal goes from right-to-left and it provides a special feature: when we reach an itemset X, all subsets of X were already discovered. As a result, this traversal can be used to filter FGs among FIs. During the traversal procedure *CGT* also filters FCIs and assigns them to the corresponding FGs, thus *CGT* outputs at the end the frequent equivalence classes. In order to filter the FGs, it must rely on the reverse pre-order strategy.

Consider the following 4×6 sample dataset: $\mathcal{D} = \{(1,\ ACDE),\ (2,\ ABCDE),\ (3,\ ABE),\ (4,\ BEF)\}$. The execution of *Talky* on dataset $\mathcal{D}$ with $min\_supp = 2$ is

[1] Due to lack of space, please refer to our paper [1] for the definitions of basic concepts.

[2] This work was partially supported by the TAMOP-4.2.2.C-11/1/KONV-2012-0001 project, itself partially funded by the European Union, co-funded by the European Social Fund. Canadian co-authors acknowledge the support of the National Science and Engineering Research Council through their respective Discovery grants.

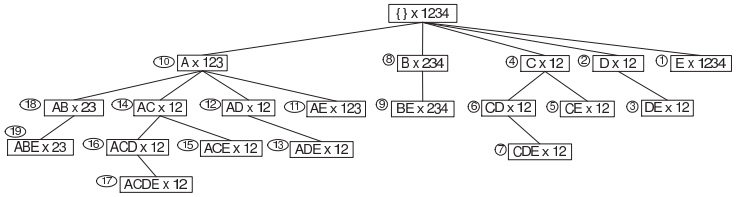

Fig. 1. Execution of *Talky* on dataset $\mathcal{D}$ with $min\_supp = 2$.

Table 1. CGT builds this table, which is actually a hash table. Key of the hash: a tidset. Value of the hash: a row of the table.

tidset	generators	eq. class members (optional)	closure	support
1234	$\emptyset$	E	E	4
234	B	BE	BE	3
123	A	AE	AE	3
23	AB	ABE	ABE	2
12	D, C	DE, CE, CD, CDE, AD, ADE, AC, ACE, ACD, $ACDE$	$ACDE$	2

shown in Figure 1. The processing order of nodes is indicated in circles. For instance, the node $C \times 12$ means that the itemset C is present in the 1st and 2nd row of the dataset, thus its support is 2.

Example. CGT builds a hash table, as depicted in Table 1. A row object represents an equivalence class. In our dataset $\mathcal{D}$, the column E is full, meaning that E is not a generator because it has a proper subset with the same support namely the empty set. Thus, the empty hash table is initialized with the line "1234;$\emptyset$; ;$\emptyset$;4". (That is, the tidset is 1234, the generators and the closure are the empty set, and the support value is 4. Eq. class members is left empty.) Then, the algorithm starts enumerating the 19 FIs of $\mathcal{D}$ using the traversal strategy of *Talky* (as seen in Figure 1). The first node is $E \times 1234$. The tidset 1234 is an existing key in the hash. E has a proper subset with the same support (the empty set), thus E is added to the "eq. class members" and "closure" fields. The "closure" column is the union of the generators and the itemsets that belong to the same equivalence class. The next FI is $D \times 12$. Since 12 is not yet in the hash, a new row is added in the hash table. The next node is $DE \times 12$. The tidset 12 is in the hash, thus DE belongs to an existing equivalence class. It has a proper generator subset, D, thus DE is added to the "eq. class members" and "closure" fields. After adding E, D, and DE, the current state of the hash table looks like this: 1st row is "1234;$\emptyset$;E;E;4"; 2nd row is "12;D;DE;DE;2". The end result is shown in Table 1.

References

1. Szathmary, L., Valtchev, P., Napoli, A., Godin, R.: Efficient Vertical Mining of Frequent Closures and Generators. In: Proc. of the 8th Intl. Symposium on Intelligent Data Analysis (IDA '09). Volume 5772 of LNCS., Lyon, France, Springer (2009) 393–404
2. Zaki, M.J., Parthasarathy, S., Ogihara, M., Li, W.: New Algorithms for Fast Discovery of Association Rules. In: Proc. of the 3rd Intl. Conf. on Knowledge Discovery in Databases. (August 1997) 283–286