Efficient Vertical Mining of Minimal Rare Itemsets

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

[1] University of Debrecen, Faculty of Informatics, Department of IT,
H-4010 Debrecen, Pf. 12, Hungary
szathmary.laszlo@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. Rare itemsets are important sort of patterns that have a wide range of practical applications, in particular, in analysis of biomedical data. Although mining rare patterns poses specific algorithmic problems, it is yet insufficiently studied. In a previous work, we proposed a levelwise approach for rare itemset mining that traverses the search space bottom-up and proceeds in two steps: (1) moving across the frequent zone until the minimal rare itemsets are reached and (2) listing all rare itemsets. As the efficiency of the frequent zone traversal is crucial for the overall performance of the rare miner, we are looking for ways to speed it up. Here, we examine the benefits of depth-first methods for that task as such methods are known to outperform the levelwise ones in many practical cases. The new method relies on a set of structural results that helps save a certain amount of computation and eventually ensures it outperforms the current levelwise procedure.

1 Introduction

Pattern mining is a basic data mining task whose aim is to uncover the hidden regularities in a set of data records, called *transactions* [1]. These regularities typically manifest themselves as repeating co-occurrences of properties, or *items*, in the transactions, i.e., item patterns. As there is a potentially huge number of patterns, quality measures are applied to filter only promising patterns, i.e., patterns of potential interest to the analyst.

Designing a faithful interestingness metric in a domain independent fashion is not realistic [2]. Indeed, without an access to the semantics of the items or another source of domain knowledge, it is impossible for the mining tool to assess the real value behind a pattern. As a simplifying hypothesis, the overwhelming majority of pattern miners chose to look exclusively on item combinations that

ISBN 978–84–695–5252–0, Universidad de Málaga (Dept. Matemática Aplicada)

are sufficiently frequent, i.e., observed in a large enough proportion of the transactions. This roughly translates the intuition that significant regularities should occur often within a dataset.

Yet such a hypothesis fails to reflect the entire variety of situations in data mining practice [3]. More precisely, it ignores some of the key factors for the success of the mining task, namely, the expectations of the analyst and further to that, her/his knowledge of the dataset and of the domain it stems from. Indeed, while an analyst with little or no knowledge of the dataset will most probably be happy with the most frequent patterns thereof, a better informed one may find them of little surprise and hence barely useful. More generally speaking, in some specific situations, frequency may be the exact opposite of pattern interestingness. The reason behind is that in these cases, the most typical item combinations from the data correspond to widely-known and well-understood phenomena, hence there is no point in presenting them to the analyst. In contrast, less frequently occurring pattern may point to unknown or poorly studied aspects of the underlying domain [3].

The above schematic description fits to a wide range of mining situations where biomedical data are involved. For instance, in pharmacovigilance, one is interested in associating the drugs taken by patients to the adverse effects the latter may present as a result (*safety signals* or *drug interactions* in the technical language of the field). To do that, a now popular way is to mine the databases of pharmacovigilance reports, where each individual case is thoroughly described, for such associations. However, as the data is accumulated throughout the years, the most frequent associations tend to translate well-known signals and interactions. The new and potentially interesting associations are less frequent and hence "hidden" behind these most often occurring combinations.

The problem of unraveling them is a non-trivial one: In [4], a method based on frequent pattern mining has been shown to only be able of dealing with a small proportion of the existing pharmacovigilance datasets. The main difficulty is that the data cannot be advantageously segmented as the new signals/interactions may appear in any record. Alternatively, the problem cannot be approached as outlier detection as a potential manifestation of a new signal need not have any exceptional characteristics. Moreover, in order for an association to be validated, it must occur in at least a given minimal number of patient records (typically, five). Yet mining all patterns with only this weak constraint results in an enormously-sized output whereby the overwhelming majority brings no new insights.

The conclusion we drew out of that study was that the not-as-frequent, or rare, patterns need to be addressed by specially designed algorithms rather than by standard frequent miners fed with lower enough support. Similar observations have been made in the pattern mining literature more than half a decade ago [3]. Since that time, a variety of methods that target non-frequent datasets have been published, most of them adapting the classical levelwise mining schema exemplified by the *Apriori* algorithm [1] to various relaxations of the frequent itemset and frequent association notions [5,6,7] (see [8] for a recent survey thereof).

In our own approach, we focus on limiting the computational effort dedicated to the traversal of irrelevant areas of the search space. In fact, as indicated above, the rare itemsets represent a band of the underlying Boolean lattice of all itemsets that is located "above" the frequent part thereof and "below" the exceptional part (itemsets that occur in a tiny number, possibly none, of transactions). Thus, in a previous paper [9], we proposed a bottom-up, levelwise approach that traverses the frequent zone of the search space either exhaustively or in a more parsimonious manner by listing uniquely frequent generator itemsets. We also provided a levelwise method for generating all rare itemsets up to the minimal frequency required by the task (could be one in the worst case).

In this paper we are looking for a more efficient manner for traversing the frequent part of the Boolean lattice. In fact, the rapidity of pinpointing the minimal rare itemsets turned out to be a dominant factor for the overall performance of the rare pattern miner. It is therefore natural to investigate manners to speed it up. Further to that idea, and breaking with the dominant levelwise algorithmic schema, we study a depth-first method. Indeed, depth-first frequent pattern miners have been shown to outperform breadth-first ones on a number of datasets. We therefore decided to check the potential benefits of the approach in the rare pattern case. To that end, we have shown a set of structural results that allows for a sound substitution within the overall rare pattern mining architecture. Our experimental results show that the new method is most of the time much faster than the previous one.

The main contribution of this paper is a new algorithm called *Walky-G* for mining minimal rare itemsets. The algorithm limits the traversal of the frequent zone to frequent generators only. This traversal is achieved through a depth-first strategy.

The remainder of the paper is organized as follows. We first recall the basic concepts of frequent/rare pattern mining and then summarize the key aspects of our own approach. Next, we present a set of structural results about the search space and the supporting structure of the depth-first traversal of the pattern space. Then, the depth-first frequent zone-traversal algorithm is described and its *modus operandi* illustrated. A comparative study of its performance to those of the current breadth-first methods is also provided. Finally, lessons learned and future work are discussed.

2 Basic Concepts

Consider the following 6×5 sample dataset: $\boldsymbol{\mathcal{D}} = \{(1,\ ABCDE), (2,\ BCD), (3,\ ABC), (4,\ ABE), (5,\ AE), (6,\ DE)\}$. Throughout the paper, we will refer to this example as **"dataset $\boldsymbol{\mathcal{D}}$"**.

Consider a set of *objects* or *transactions* $\mathcal{O} = \{o_1, o_2, \ldots, o_m\}$, a set of *attributes* or *items* $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$, and a relation $\mathcal{R} \subseteq \mathcal{O} \times \mathcal{A}$. A set of items is called an *itemset*. Each transaction has a unique identifier (*tid*), and a set of transactions is called a *tidset*. The tidset of all transactions sharing a given itemset X is its *image*, denoted by $t(X)$. The *length* of an itemset X is $|X|$, whereas an itemset of length i is called an i-itemset. The (absolute) *support* of an itemset X, denoted by $supp(X)$, is the size of its image, i.e. $supp(X) = |t(X)|$.

A lattice can be separated into two segments or zones through a user-provided "minimum support" threshold, denoted by *min_supp*. Thus, given an itemset X, if $supp(X) \geq min\_supp$, then it is called *frequent*, otherwise it is called *rare* (or *infrequent*). In the lattice in Figure 1, the two zones corresponding to a support threshold of 2 are separated by a solid line. The rare itemset family and the corresponding lattice zone is the target structure of our study.

Definition 1. X subsumes Z*, iff* $X \supset Z$ *and* $supp(X) = supp(Z)$ [10].

Definition 2. *An itemset* Z *is a* generator *if it has no proper subset with the same support.*

Generators are also known as free-sets [11] and have been targeted by dedicated miners [12].

Property 1. Given $X \subseteq \mathcal{A}$, if X is a generator, then $\forall Y \subseteq X$, Y is a generator, whereas if X is not a generator, $\forall Z \supseteq X$, Z is not a generator [13].

Proposition 1. *An itemset* X *is a generator iff* $supp(X) \neq min_{i \in X}(supp(X \setminus \{i\}))$ [14].

Each of the frequent and rare zones is delimited by two subsets, the maximal elements and the minimal ones, respectively. The above intuitive ideas are formalized in the notion of a border introduced by Mannila and Toivonen in [15]. According to their definition, the maximal frequent itemsets constitute the *positive border* of the frequent zone[1] whereas the minimal rare itemsets form the *negative border* of the same zone.

Definition 3. *An itemset is a* maximal frequent itemset *(MFI) if it is frequent but all its proper supersets are rare.*

Definition 4. *An itemset is a* minimal rare itemset *(mRI) if it is rare but all its proper subsets are frequent.*

The levelwise search yields as a by-product all mRIs [15]. Hence we prefer a different optimization strategy that still yields mRIs while traversing only a subset of the frequent zone of the Boolean lattice. It exploits the minimal generator status of the mRIs. By Property 1, frequent generators (FGs) can be traversed in a levelwise manner while yielding their negative border as a by-product. It is enough to observe that mRIs are in fact generators:

Proposition 2. *All minimal rare itemsets are generators* [9].

[1] The frequent zone contains the set of frequent itemsets.

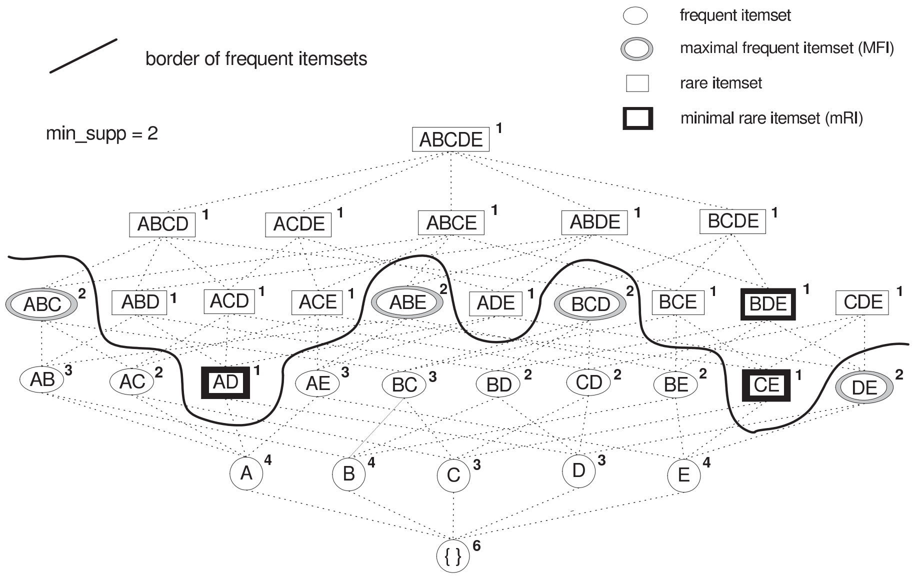

Fig. 1. The powerset lattice of dataset $\mathcal{D}$.

Finding Minimal Rare Itemsets in a Levelwise Manner

As pointed out by Mannila and Toivonen [15], the easiest way to reach the negative border of the frequent itemset zone, i.e., the mRIs, is to use a levelwise algorithm such as *Apriori*. Indeed, albeit a frequent itemset miner, *Apriori* yields the mRIs as a by-product.

Apriori-Rare [9] is a slightly modified version of *Apriori* that retains the mRIs. Thus, whenever an i-long candidate survives the frequent $i-1$ subset test, but proves to be rare, it is kept as an mRI.

MRG-Exp [9] produces the same output as *Apriori-Rare* but in a more efficient way. Following Proposition 2, *MRG-Exp* avoids exploring all frequent itemsets: instead, it looks after frequent generators only. In this case mRIs, which are rare generators as well, can be filtered among the negative border of the frequent generators. The output of *MRG-Exp* is identical to the output of *Apriori-Rare*, i.e. both algorithms find the set of mRIs.

3 Finding Minimal Rare Itemsets in a Depth-First Manner

Eclat [16] was the first FI-miner to combine the vertical encoding with a depth-first traversal of a tree structure, called IT-tree, whose nodes are $X \times t(X)$ pairs. *Eclat* traverses the IT-tree in a depth-first manner in a pre-order way, from left-to-right [16,17] (see Figure 2).

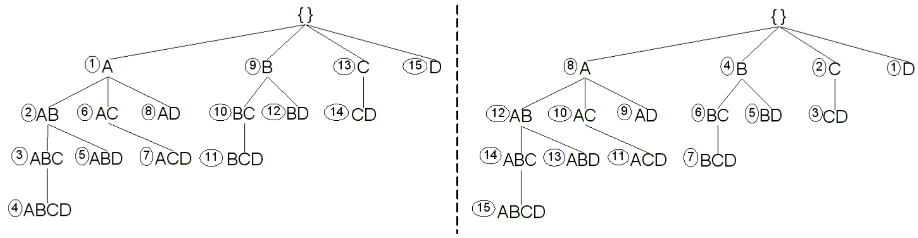

Fig. 2. Left: pre-order traversal with *Eclat*; **Right:** reverse pre-order traversal with *Eclat.* The direction of traversal is indicated in circles

3.1 Talky-G

Talky-G [18] is a vertical FG-miner following a depth-first traversal of the IT-tree and a right-to-left order on sibling nodes. *Talky-G* applies an inclusion-compatible traversal: it goes down the IT-tree while listing sibling nodes from right-to-left and not the other way round as in *Eclat.*

The authors of [19] explored that order for mining calling it *reverse pre-order*. They observed that for any itemset X its subsets appear in the IT-tree in nodes that lay either higher on the same branch as $(X, t(X))$ or on branches to the right of it. Hence, depth-first processing of the branches from right-to-left would perfectly match set inclusion, i.e., all subsets of X are met before X itself. While the algorithm in [19] extracts the so-called non-derivable itemsets, *Talky-G* uses this traversal to find the set of frequent generators. See Figure 2 for a comparison of *Eclat* and its “reversed” version.

3.2 Walky-G

In this subsection we present the algorithm *Walky-G*, which is the main contribution of this paper. Since *Walky-G* is an extension of *Talky-G*, we also present the latter algorithm at the same time. *Walky-G*, in addition to *Talky-G*, retains rare itemsets and checks them for minimality.

Hash structure. In *Walky-G* a hash structure is used for storing the already found frequent generators. This hash, called $fgMap$, is a simple dictionary with key/value pairs, where the key is an itemset (a frequent generator) and the value is the itemset's support.[2] The usefulness of this hash is twofold. First, it allows a quick look-up of the proper subsets of an itemset with the same support, thus the generator status of a frequent itemset can be tested easily (see Proposition 1). Second, this hash is also used to look-up the proper subsets of a minimal rare candidate. This way rare but non-minimal itemsets can be detected efficiently.

Pseudo code. Algorithm 1 provides the main block of *Walky-G*. First, the

[2] In our implementation we used the `java.util.HashMap` class for $fgMap$.

Algorithm 1 (main block of Walky-G):

```
// for quick look-up of (1) proper subsets with the same support
// and (2) one-size smaller subsets:
fgMap ← ∅   // key: itemset (frequent generator); value: support

root.itemset ← ∅   // root is an IT-node whose itemset is empty
// the empty set is included in every transaction:
root.tidset ← {all transaction IDs}
fgMap.put(∅, |O|)   // the empty set is an FG with support 100%
loop over the vertical representation of the dataset (attr) {
      if (min_supp ≤ attr.supp < |O|) {
            // |O| is the total number of objects in the dataset
            root.addChild(attr)   // attr is frequent and generator
      }
      if (0 < attr.supp < min_supp) {
            saveMri(attr)   // attr is a minimal rare itemset
      }
}
loop over the children of root from right-to-left (child) {
      saveFg(child)   // the direct children of root are FGs
      extend(child)   // discover the subtree below child
}
```

IT-tree is initialized, which involves the creation of the root node, representing the empty set (of 100% support, by construction). *Walky-G* then transforms the layout of the dataset in vertical format, and inserts below the root node all 1-long frequent itemsets. Such a set is an FG whenever its support is less than 100%. Rare attributes (whose support is less than min_supp) are minimal rare itemsets since all their subsets (in this case, the empty set) are frequent. Rare attributes with support 0 are not considered.

The `saveMri` procedure processes the given minimal rare itemset by storing it in a database, by printing it to the standard output, etc. At this point, the dataset is no more needed since larger itemsets can be obtained as unions of smaller ones while for the images intersection must be used.

The `addChild` procedure inserts an IT-node under a node. The `saveFg` procedure stores a given FG with its support value in the hash structure $fgMap$.

In the core processing, the `extend` procedure (see Algorithm 2) is called recursively for each child of the root in a right-to-left order. At the end, the IT-tree contains all FGs. Rare itemsets are verified during the construction of the IT-tree and minimal rare itemsets are retained. The `extend` procedure discovers all FGs in the subtree of a node. First, new FGs are tentatively generated from the right siblings of the current node. Then, certified FGs are added below the current node and later on extended recursively in a right-to-left order.

The `getNextGenerator` function (see Algorithm 3) takes two nodes and returns a new FG, or “null” if no FG can be produced from the input nodes. In addition, this method tests rare itemsets and retains the minimal ones. First, a candidate node is created by taking the union of both itemsets and the in-

Algorithm 2 ("extend" procedure):

Method: extend an IT-node recursively (discover FGs in its subtree)
Input: an IT-node (*curr*)

```
loop over the right siblings of curr from left-to-right (other) {
      generator ← getNextGenerator(curr, other)
      if (generator ≠ null) then curr.addChild(generator)
}
loop over the children of curr from right-to-left (child) {
      saveFg(child)   // child is a frequent generator
      extend(child)   // discover the subtree below child
}
```

tersection of their respective images. The input nodes are thus the candidate's *parents*. Then, the candidate undergoes a frequency test (test 1). If the test fails then the candidate is rare. In this case, the minimality of the rare itemset *cand* is tested. If all its one-size smaller subsets are present in $fgMap$ then *cand* is a minimal rare generator since all its subsets are FGs (see Property 1). From Proposition 2 it follows that an mRG is an mRI too, thus *cand* is processed by the `saveMri` procedure. If the frequency test was successful, the candidate is compared to its parents (test 2): if its tidset is equivalent to a parent tidset, then the candidate cannot be a generator. Even with both outcomes positive, an itemset may still not be a generator as a subsumed subset may lay elsewhere in the IT-tree. Due to the traversal strategy in *Walky-G*, all generator subsets of the current candidate are already detected and the algorithm has stored them in $fgMap$ (see the `saveFg` procedure). Thus, the ultimate test (test 3) checks whether the candidate has a proper subset with the same support in $fgMap$. A positive outcome disqualifies the candidate.

This last test (test 3) is done in Algorithm 4. First, one-size smaller subsets of *cand* are collected in a list. The two parents of *cand* can be excluded since *cand* was already compared to them in test 2 in Algorithm 3. If the support value of one of these subsets is equal to the support of *cand*, then *cand* cannot be a generator. Note that when the one-size smaller subsets are looked up in $fgMap$, it can be possible that a subset is missing from the hash. It means that the missing subset was tested before and turned out to subsume an FG, thus the subset was not added to $fgMap$. In this case *cand* has a non-FG subset, thus *cand* cannot be a generator either (by Property 1).

Candidates surviving the final test in Algorithm 3 are declared FG and added to the IT-tree. An unsuccessful candidate X is discarded which ultimately prevents any itemset Y having X as a prefix to be generated as candidate and hence substantially reduces the overall search space. When the algorithm stops, all frequent generators (and *only* frequent generators) are inserted in the IT-tree *and* in the $fgMap$ structure. Furthermore, upon the termination of the algorithm, all minimal rare itemsets have been found. For a running example, see Figure 3.

Algorithm 3 ("getNextGenerator" function):

Method: create a new frequent generator *and* filter minimal rare itemsets
Input: two IT-nodes (*curr* and *other*)
Output: a frequent generator or null

```
cand.itemset ← curr.itemset ∪ other.itemset
cand.tidset ← curr.tidset ∩ other.tidset
if (cardinality(cand.tidset) < min_supp)   // test 1: frequent?
{    // now cand is an mRI candidate; let us test its minimality:
     if (all one-size smaller subsets of cand are in fgMap) {
          saveMri(cand)   // cand is an mRI, save it
     }
     return null   // not frequent
}
// else, if it is frequent; test 2:
if ((cand.tidset = curr.tidset) or (cand.tidset = other.tidset)) {
     return null   // not a generator
}
// else, if it is a potential frequent generator; test 3:
if (candSubsumesAnFg(cand)) {
     return null   // not a generator
}
// if cand passed all the tests then cand is a frequent generator
return cand
```

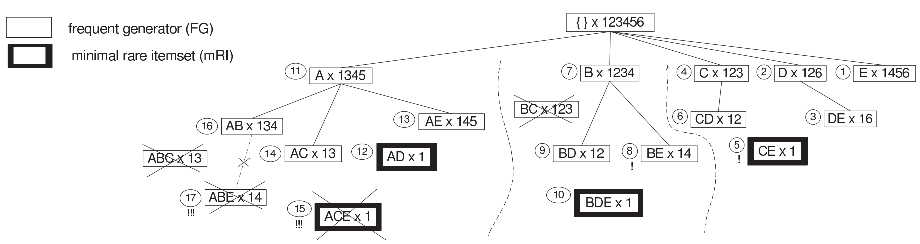

Fig. 3. The IT-tree built during the execution of *Walky-G* on dataset $\mathcal{D}$ with $min\_supp = 2$ (33%). Notice the two special cases: ACE is not an mRI because of CE; ABE is not an FG because of BE.

4 Experimental Results

In our experiments, we compared *Walky-G* against *Apriori-Rare* [9] and *MRG-Exp* [9]. The algorithms were implemented in Java in the Coron platform [20].[3] The experiments were carried out on a bi-processor Intel Quad Core Xeon 2.33 GHz machine running under Ubuntu GNU/Linux with 4 GB of RAM. All times reported are real, wall clock times as obtained from the Unix *time* command

[3] http://coron.loria.fr

Algorithm 4 ("candSubsumesAnFg" function):

Method: verify if *cand* subsumes an already found FG
Input: an IT-node (*cand*)

```
subsets ← {one-size smaller subsets of cand minus the two parents}
loop over the elements of subsets (ss) {
    if (ss is stored in fgMap) {
        stored_support ← fgMap.get(ss)   // get the support of ss
        if (stored_support = cand.support) {
            return true   // case 1: cand subsumes an FG
        }
    }
    else   // if ss is not present in fgMap
    {   // case 2: cand has a non-FG subset ⇒ cand is not an FG either
        return true
    }
}
return false   // if we get here then cand is an FG
```

between input and output. For the experiments we have used the following datasets: T20I6D100K, C20D10K, C73D10K, and MUSHROOMS. The T20[4] is a sparse dataset, constructed according to the properties of market basket data that are typical weakly correlated data. The C20 and C73 are census datasets from the PUMS sample file, while the MUSHROOMS[5] describes mushrooms characteristics. The last three are highly correlated datasets.

The execution times of the three algorithms are illustrated in Table 1. The table also shows the number of frequent itemsets, the number of frequent generators, the proportion of the number of FGs to the number of FIs, and the number of minimal rare itemsets. The last column shows the number of mRIs whose support values exceed 0.

The T20 synthetic dataset mimics market basket data that are typical sparse, weakly correlated data. In this dataset, the number of FIs is small and nearly all FIs are generators. Thus, *MRG-Exp* works exactly like *Apriori-Rare*, i.e. it has to explore almost the same search space. Though *Walky-G* needs to explore a search space similar to *Apriori-Rare*'s, it can perform much better due to its depth-first traversal.

In datasets C20, C73, and MUSHROOMS, the number of FGs is much less than the total number of FIs. Hence, *MRG-Exp* and *Walky-G* can take advantage of exploring a much smaller search space than *Apriori-Rare*. Thus, *MRG-Exp* and *Walky-G* perform much better on dense, highly correlated data. For example, on MUSHROOMS at $min\_supp = 10\%$, *Apriori-Rare* needs to extract 600,817 FIs, while *MRG-Exp* and *Walky-G* extract 7,585 FGs only. This means that *MRG-Exp* and *Walky-G* reduce the search space of *Apriori-Rare* to 1.26%! The

[4] http://www.almaden.ibm.com/software/quest/Resources/
[5] http://kdd.ics.uci.edu/

Table 1. Response times of Apriori-Rare, MRG-Exp, and Walky-G.

min_supp	execution time (sec.)			# FIs	# FGs	$\frac{\#FGs}{\#FIs}$	# mRIs
	Apriori-Rare	MRG-Exp	**Walky-G**				(support > 0)
T20I6D100K							
10%	3.25	3.24	**1.61**	7	7	100.00%	907
0.75%	30.22	30.92	**11.80**	4,710	4,710	100.00%	211,561
0.5%	49.30	48.82	**15.89**	26,836	26,305	98.02%	268,589
0.25%	115.35	117.11	**33.47**	155,163	149,447	96.32%	534,088
C20D10K							
30%	21.92	5.49	**0.57**	5,319	967	18.18%	226
20%	56.43	9.70	**0.62**	20,239	2,671	13.20%	376
10%	157.09	18.27	**0.77**	89,883	9,331	10.38%	837
5%	366.34	28.35	**0.93**	352,611	23,051	6.54%	1,867
2%	878.93	40.77	**1.47**	1,741,883	57,659	3.31%	7,065
C73D10K							
95%	35.97	6.97	**0.84**	1,007	121	12.02%	1,622
90%	453.93	48.65	**0.90**	13,463	1,368	10.16%	1,701
85%	1,668.19	117.62	**0.95**	46,575	3,513	7.54%	1,652
MUSHROOMS							
40%	3.24	1.77	**0.50**	505	153	30.30%	251
30%	9.39	3.09	**0.51**	2,587	544	21.03%	402
15%	160.88	8.32	**0.66**	99,079	3,084	3.11%	1,741
10%	676.53	13.22	**0.76**	600,817	7,585	1.26%	2,916

advantages of the depth-first approach of *Walky-G* is more spectacular on dense datasets: the execution times, with the exception of one case in Table 1, are always below 1 second.

5 Conclusion and Future Work

We presented an approach for rare itemset mining from a dataset that splits the problem into two tasks. Our new algorithm, *Walky-G*, limits the traversal of the frequent zone to frequent generators *only*. This traversal is achieved through a depth-first strategy. Experimental results prove the interest of our method not only on dense, highly correlated datasets, but on sparse ones too. Our approach breaks with the dominant levelwise algorithmic schema and shows that it outperforms its current levelwise competitors.

References

1. Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., Verkamo, A.I.: Fast discovery of association rules. In: Advances in knowledge discovery and data mining. American Association for Artificial Intelligence (1996) 307–328
2. Hand, D., Mannila, H., Smyth, P.: Principles of Data Mining. The MIT Press, Cambridge (MA) (2001)
3. Weiss, G.: Mining with rarity: a unifying framework. SIGKDD Explor. Newsl. **6**(1) (2004) 7–19

4. Hacene, M.R., Toussaint, Y., Valtchev, P.: Mining safety signals in spontaneous reports database using concept analysis. In: Proc. 12th Conf. on AI in Medicine, AIME 2009. Volume 5651 of Lecture Notes in Computer Science. (2009) 285–294
5. Liu, B., Hsu, W., Ma, Y.: Mining Association Rules with Multiple Minimum Supports. In: Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '99), New York, NY, USA, ACM Press (1999) 337–341
6. Yun, H., Ha, D., Hwang, B., Ryu, K.: Mining association rules on significant rare data using relative support. Journal of Systems and Software **67**(3) (2003) 181–191
7. Koh, Y., Rountree, N.: Finding Sporadic Rules Using Apriori-Inverse. In: Proceedings of the 9th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining (PAKDD '05), Hanoi, Vietnam. Volume 3518 of Lecture Notes in Computer Science., Springer (May 2005) 97–106
8. Koh, Y.S., Rountree, N.: Rare Association Rule Mining and Knowledge Discovery: Technologies for Infrequent and Critical Event Detection. Information Science Reference - Imprint of: IGI Publishing, Hershey, PA (2009)
9. Szathmary, L., Napoli, A., Valtchev, P.: Towards Rare Itemset Mining. In: Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI '07). Volume 1., Patras, Greece (Oct 2007) 305–312
10. Zaki, M.J., Hsiao, C.J.: CHARM: An Efficient Algorithm for Closed Itemset Mining. In: SIAM International Conference on Data Mining (SDM' 02). (Apr 2002) 33–43
11. Calders, T., Rigotti, C., Boulicaut, J.F.: A Survey on Condensed Representations for Frequent Sets. In Boulicaut, J.F., Raedt, L.D., Mannila, H., eds.: Constraint-Based Mining and Inductive Databases. Volume 3848 of Lecture Notes in Computer Science., Springer (2004) 64–80
12. Li, J., Li, H., Wong, L., Pei, J., Dong, G.: Minimum Description Length Principle: Generators Are Preferable to Closed Patterns. In: AAAI, AAAI Press (2006) 409–414
13. Kryszkiewicz, M.: Concise Representations of Association Rules. In: Proceedings of the ESF Exploratory Workshop on Pattern Detection and Discovery. (2002) 92–109
14. Bastide, Y., Taouil, R., Pasquier, N., Stumme, G., Lakhal, L.: Mining Frequent Patterns with Counting Inference. SIGKDD Explor. Newsl. **2**(2) (2000) 66–75
15. Mannila, H., Toivonen, H.: Levelwise Search and Borders of Theories in Knowledge Discovery. Data Mining and Knowledge Discovery **1**(3) (1997) 241–258
16. Zaki, M.J., Parthasarathy, S., Ogihara, M., Li, W.: New Algorithms for Fast Discovery of Association Rules. In: Proceedings of the 3rd International Conference on Knowledge Discovery in Databases. (August 1997) 283–286
17. Zaki, M.J.: Scalable Algorithms for Association Mining. IEEE Transactions on Knowledge and Data Engineering **12**(3) (2000) 372–390
18. 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
19. Calders, T., Goethals, B.: Depth-first non-derivable itemset mining. In: Proceedings of the SIAM International Conference on Data Mining (SDM '05), Newport Beach, USA. (Apr 2005)
20. Szathmary, L.: Symbolic Data Mining Methods with the Coron Platform. PhD Thesis in Computer Science, Univ. Henri Poincaré – Nancy 1, France (Nov 2006)