Characterizing Transactional Databases for Frequent Itemset Mining

Christian Lezcano* Marta Arias†

Abstract

This paper presents a study of the characteristics of transactional databases used in frequent itemset mining. Such characterizations have typically been used to benchmark and understand the data mining algorithms working on these databases. The aim of our study is to give a picture of how diverse and representative these benchmarking databases are, both in general but also in the context of particular empirical studies found in the literature. Our proposed list of metrics contains many of the existing metrics found in the literature, as well as new ones. Our study shows that our list of metrics is able to capture much of the datasets' inner complexity and thus provides a good basis for the characterization of transactional datasets. Finally, we provide a set of representative datasets based on our characterization that may be used as a benchmark safely.

1 Introduction

Since the introduction of Frequent Itemset Mining (*FIM*) and its early algorithms, a huge number of algorithms have been proposed [1, 14, 18, 22, 23] (see Fournier-Viger et al. [9] for a more detailed review of the latest *FIM* approaches); in fact, itemset mining and the closely related association rule mining have been arguably the hottest topic within the field of data mining for years. With the appearance of competing itemset mining algorithms comes the need to understand their strengths and weaknesses. Natural questions arise: what algorithm is the fastest for one particular dataset[1]? What is the best algorithm? Or more realistically: what algorithms work best for what types of datasets?

In an attempt to answer these questions, authors set up and run empirical studies (what we call *algorithm benchmarking* here). In data mining algorithm benchmarking, one uses a set of datasets as the basis for comparison (a *benchmark*), and applies all competing algorithm candidates to the benchmark in order to establish what algorithm is the fastest, uses less memory or gives more accurate results. In order to make the comparison fair, one should use as many datasets as possible, and these should be as diverse as possible. In other words, the benchmark should be a representative sample of what is to be expected when applying the algorithms in real-world scenarios.

The question of whether a benchmark (that is, a set of databases used for comparing algorithms) is representative or not is a difficult one to answer. Our proposed approach in this work is to characterize each dataset with the computation of several metrics, each capturing a different aspect of the datasets' complexity and structure. Trivial metrics are for example the number of transactions in the database, and more complex ones are the ones establishing, for example, their *density* [12]. Once one has a description of each dataset by means of a vector of metrics, one can see how the different metrics' values are distributed. If they cover a wide range collectively, then the benchmark is a good candidate for being representative. If, on the other hand, values are all clumped together, then the benchmark is probably not a good one to use.

Another use for the characterization of databases is to establish connections between database characteristics and the performance of particular data mining algorithms, so that one could for example make claims such as "algorithm A works well for databases that have many transactions and are dense, but algorithm B works better if the database is sparse and small."

In fact, there are several works that do precisely this, namely, establishing connections between dataset characteristics and algorithm performance [11, 24]. We detail these works in the next section.

An important tool that we exploit in this paper is the *FIMI* public repository, where *FIM* algorithms and benchmark datasets were made publicly accessible to the *FIM* community. Since its introduction by Goethals and Zaki [11], this repository serves as a standard set of tools to benchmarking algorithmic approaches. It has been very successful in the sense that most benchmarking studies use datasets from this repository. Another public repository of transactional datasets is the *SPMF* open-source data mining mining library. This repository is constantly updated and provides a greater number of *FIM* implementations and datasets

*Computer Science Department, Universitat Politècnica de Catalunya, Spain, clezcano@cs.upc.edu

†Computer Science Department, Universitat Politècnica de Catalunya, Spain, marias@cs.upc.edu

[1]We use dataset and database interchangeably.

[8]. We believe that is paramount to characterize the transactional databases present in both repositories. For that reason, we use metrics found in the literature and others of our own in order to understand their nature and evaluate their representativeness.

Another area where characterizing databases may be of use is in synthetic database generation. When attempting to generate synthetic data, one needs some control over the generated data. This control may come in the shape of characterizing the data generated, for example, by means of metrics similar to the ones we propose here.

Finally, we believe that taking into account the characterization of databases when making claims about an algorithm's performance over competitors is important in order to give enough support to such claims. We recognize the fact that this may be out of the scope of some work where the focus is on algorithm development. To make the life of such developers easier, we propose a benchmark that covers the full spectrum of values found in our study as a "minimum representative benchmark" (MRB), so that authors that use our proposed MRB have some guarantees that the databases used are representative. Naturally, if in the future new metrics emerge, the list may have to be revised, so we propose this as an evolving MRB. To summarize, the contributions in this paper are:

1. We provide a comprehensive list of metrics used for database characterization
2. We evaluate the metrics mentioned in the item above over publicly available and commonly used databases used for frequent itemset mining algorithms
3. We define and study benchmark representativeness of existing works
4. We propose a minimum representative benchmark, i.e., a benchmark whose representativeness is guaranteed according to the metrics included here

The organization of this paper is as follows. Section 2 describes the related work which is divided into three parts. Section 3 presents the definition of the metrics used in the characterization of transactional datasets. Then, in Section 4 the experiments and evaluations carried out in this work are presented. Finally, Section 5 explains the conclusion and future work.

2 Related work

On the connections between database characteristics and algorithm's performance. To the best of our knowledge, the work by Zheng et al. [24] was the first one to note the relevance of data characteristics over an algorithm's performance. The authors show that algorithms that appear better suited for some specific dataset properties do not respond in the same way for other classes of datasets. Specifically, they report that algorithms measured over synthetic datasets behave very differently when run over real datasets. The same authors explain that the reason for this outcome might be that the algorithms were fine-tuned for the synthetic dataset characteristics used in their experiment which caused to perform poorly on the real-life datasets. Since then there has been awareness of the importance of benchmarking methods over databases with different set of dataset characteristics. Goethals and Zaki [11] address the problem claimed by Zheng et al. [24] and introduce the *FIMI* repository with much success.

On algorithm benchmarking. The work of Zimmermann [25] points out several existing issues in the evaluation of frequent itemset mining algorithms. Here, the author denounces the lack of the necessary diversity of characteristics to fully understand the algorithms' strengths and limitations. The author notes that it is not the number of datasets utilized in measuring the quality of an algorithm that matters, but it is evaluating an algorithm with datasets with a variety of characteristics representing real world scenarios. On the other hand, it is also noted by the aforementioned author that simply adding more datasets to a repository does not entail that more different characteristics are being added. Briefly, the problem above motivates the use of a collection of real datasets or artificial dataset generators that emulate the whole spectrum of genuine characteristics found in real-life databases. In the same way, it is explained that every dataset utilized in benchmarking should be comprised of characteristics which emerge from a real-life process. This means that merely generating new datasets under randomly selected characteristics can not be considered appropriate since these datasets do not reflect real-life behavior. Our work is clearly motivated by the criticisms made in this work.

On database characterization. A central property of transactional databases is their density. The best intuitive notion of dataset density is given by Gouda and Zaki [12] who explain that having long frequent itemsets at high levels of support is what makes a dataset dense. This is why frequent mining algorithms typically adapt their method tactics according to this feature since it is more difficult to work through a dense dataset than a sparse one.

Among the works proposing new metrics for the evaluation of transactional databases properties, Gouda and Zaki [12] is the first one, to the best of our knowl-

Table 1: Characteristics of benchmarking datasets.

	Dataset	DS	AS	ATS	MTS	$F1$ (%)	GGD (%)	H_1	H_2
1.	forests	246	206	61.26	162	29.73	89.88	7.07	13.24
2.	bogPlants	377	315	14.65	39	4.65	16.57	6.56	11.56
3.	chess	3196	75	37.00	37	49.33	93.05	5.81	10.57
4.	foodmart	4141	1559	4.42	14	0.28	3.18	10.55	15.21
5.	mushroom	8124	119	23.00	23	19.33	50.24	5.95	10.61
6.	pumsb	49046	2113	74.00	74	3.50	23.93	7.67	14.17
7.	pumsbStar	49046	2088	50.48	63	2.42	22.15	7.76	14.19
8.	bmsWebview1	59602	497	2.51	267	0.51	51.90	7.85	14.84
9.	connect	67557	129	43.00	43	33.33	82.69	6.12	11.18
10.	bmsWebview2	77512	3340	4.62	161	0.14	12.95	10.46	18.07
11.	belgiumRetail	88162	16470	10.31	76	0.06	2.65	11.27	20.45
12.	skin	245057	12	3.99	4	33.30	84.85	3.36	5.01
13.	accidents	340183	468	33.81	51	7.22	42.84	6.49	11.91
14.	onlineRetail	541909	2604	4.36	8	0.17	0.53	8.91	12.59
15.	recordLink	574913	27	10.00	10	37.04	78.63	3.80	6.43
16.	kosarak	990002	41270	8.09	2498	0.02	3.89	10.43	
17.	kddcup99	1000000	135	16.00	16	11.85	38.93	5.04	8.54
18.	pamp	1000000	82	23.93	26	29.19	86.57	5.48	9.85
19.	uscensus	1000000	316	48.00	48	15.19	83.92	7.14	12.99
20.	powerc	1040000	125	7.00	7	5.60	48.97	4.25	7.07
21.	chainstore	1112949	46086	7.23	170	0.02	2.84	12.96	

edge, to introduce a classification metric for dataset density which is based mainly on the positive border length distribution. Essentially, the authors study the shape of the positive border distribution cut off at specific levels of support and classify datasets into four distinct types. Even though this work sheds some light on the characterization of datasets, it does not give any notion of how the proposed classification behaves over different levels of support.

Other works that use positive borders for the characterization of databases are Ramesh et al. [19, 20]. Instead of using them to expand our understanding of dataset properties, they focus on the problem of synthetic database generation. For this, multiple levels of positive borders distributions are extracted from an original dataset and transferred to a synthetic one afterwards.

Palmerini et al. [16] propose new measures to learn a dataset density based on the information theoretic concept of entropy and the average support of frequent itemsets. This work uses entropy to take a glance at the database density without going though the inspection of the entire database which is convenient in helping algorithms to decide the best action to take at runtime.

Flouvat et al. [6, 7] introduce a dataset classification which follows the idea of positive borders of Gouda and Zaki [12] described above. The difference is that Flouvat et al. [7] classifiy dataset density considering negative border length distribution along with that of positive borders. The authors find that negative borders provide extra information which allows a finer understanding of datasets. Concretely, an algorithm's performance is affected by how separated the mean distance between both borders is.

All of the metrics mentioned in these related works are included in our list of proposed metrics, as well as some others. The next section details the different metrics and related concepts considered in our study.

3 Definition of metrics

We propose to carry out different measurements on the *FIMI* and *SPMF* benchmark datasets utilizing metrics found in the literature and new ones of our own.

We start by defining basic properties of a transactional database such as ***dataset size (DS)*** which is simply its number of transactions, ***average transaction size (ATS)*** as the name suggests is the average of all transaction sizes, and ***maximum transaction size (MTS)*** is the maximum size value of all transactions.

Most of the following metrics are based on *FIM*. Let I be a finite set of different elements called items and

let $|I|$ represent its size. In other words, I represents the database's alphabet and hereafter we call the ***alphabet size*** as ***(AS)***. Any subset of I is denoted as an itemset X. In particular, an itemset with size k is regarded as a k-itemset. Let D be a transactional database of size $|D|$ in which each transaction is represented by an itemset. Duplication of transactions may exist; thus, transactions are differentiated via identifiers.

The support of an itemset $sup(X)$ is defined as the cardinality of the set of transactions in D in which X is a subset. X is considered frequent if its support is greater than or equal to a minimum support $minsup$ defined by the user, i.e., $sup(X) \geq minsup$. This allows to define $FI(minsup)$ or simply FI as the set of all frequent itemsets with support greater or equal to $minsup$.

Durand and Quafafou [5] present an intuitive view of *frequent itemset* borders where the positive border is the set of its maximal frequent itemsets (Equation 3.1) and the negative border is the set of minimal infrequent itemsets (Equation 3.2).

$$Bd^{+}(FI) = \{X \in FI \mid \forall Y \supset X, Y \notin FI\} \tag{3.1}$$

$$Bd^{-}(FI) = \{X \in 2^{I} \setminus FI \mid \forall Y \subset X, Y \in FI\} \tag{3.2}$$

Maximum singleton support (MSS) [20] is the maximum over all singleton supports, i.e., $MSS = \max_{\forall X \in I} sup(X)$. This metric provides an upper bound of the support of any itemset and therefore constitutes an important parameter. This metric also guides us in choosing appropriate thresholds of min support values. That is, S is the sequence of $minsup$ values given by the user in a itemset mining operation. Namely, $S = \langle s_1, s_2, s_3, \ldots, s_m \rangle$ where $0 < s_k < s_{k+1}$ and $s_m \leq MSS$.

Maximum cardinality difference (MCD) is defined to approximate how the number of frequent itemsets is distributed throughout the range of possible supports S defined above. That is, we defined MCD as the area under the curve defined by the set of ordered pairs $\{(s, |FI(s)|) \mid \forall s \in S\}$. After normalization, a low MCD value indicates that the greater percentage of frequent itemsets concentrates at the lowest levels of support whereas a higher value suggests the number of frequent itemsets does not variate abruptly throughout the different support levels S.

Similar to MCD, ***positive border cardinality (PBC)*** is the area under the curve given by the set of ordered pairs $\{(s, |Bd^{+}(FI(s))|) \mid \forall s \in S\}$. Here, we are interested in knowing how the cardinality of the positive border set (Equation 3.1) changes over the different levels of support.

Likewise, ***negative border cardinality (NBC)*** is based on the same formulation as PBC, but instead it utilizes the negative border set (Equation 3.2) to define the set of points $\{(s, |Bd^{-}(FI(s))|) \mid \forall s \in S\}$.

Gaifman graph density (GGD) is another metric we use to approximate the dataset density. Here, an undirected graph $G = (V, E)$ called Gaifman graph is built from a transactional database D. Its vertices are items (i.e. $V = I$) and two items share an edge if they appear in at least a transaction together. Then, we compute the density of the Gaifman graph G as the ratio of the cardinality of its set of edges $|E|$ to the maximum number of edges, that is $GGD = \frac{2\ |E|}{|I|\ (|I|-1)}$.

Palmerini et al. [16] propose three metrics. The first one, ***fraction of 1s (F1)***, represents the dataset D as a binary matrix D' where every row is deemed as a transaction of D and every column as an item $X \in I$. An element in D' is marked by 1 depending on whether item X appears in the transaction, and 0 otherwise. The fraction of 1s is calculated as the ratio of the number of 1s in D' to its number of elements, i.e., $F1 = \frac{cardinality\ of\ 1s}{|D|\ |I|}$.

The second one, FI average support, as its name suggests takes the result of a mining process—$FI(minsup)$ along with every frequent itemset support—and measures the distance between the average support of all FI itemsets and $minsup$. We take this idea, however, we here consider ***average support distance (ASD)*** as the area under the curve formed by the points $\{(s, \gamma_s) \mid \forall s \in S\}$ where $\gamma_s = \frac{1}{|FI(s)|} \sum_{\forall X \in FI(s)} sup(X)$. Hence, it is assumed that a database is denser the greater its ASD value is.

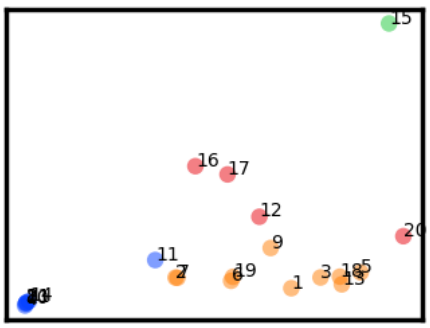

Figure 1: Scatter plot of our 21 datasets using metrics ASD (x axis) and PBC (y axis). The four clusters found by `K-Means` are color-marked.

We define ***FI average length (FAL)*** using a procedure similar to ASD. However, instead of considering the average support of all FI itemsets, here FAL computes the area under the curve using the average length of all FI itemsets for every $minsup$ defined by the user.

Table 2: Characteristics of benchmarking datasets which are calculated using the levels of support S defined for each dataset.

	Dataset	MSS (%)	Levels of support S (%)	MCD	ASD	FAL	PBC	PBL
1.	forests	93.09	$\langle 30, 40, 50, \ldots, 90\rangle$	519.69	3805.10	242.07	544.74	270.20
2.	bogPlants	65.25	$\langle 10, 20, 30, \ldots, 60\rangle$	699.62	2151.31	81.33	867.58	91.87
3.	chess	99.97	$\langle 20, 30, 40, \ldots, 90\rangle$	656.69	4222.48	544.58	870.26	681.04
4.	foodmart	0.60	$\langle 0.1, 0.2, \ldots, 0.6\rangle$	18.36	0.21	0.50	18.36	0.50
5.	mushroom	100.00	$\langle 10, 20, 30, \ldots, 90\rangle$	599.48	4796.45	301.26	1022.85	396.58
6.	pumsb	99.79	$\langle 50, 60, 70, 80, 90\rangle$	634.85	2944.61	349.69	770.19	391.60
7.	pumsbStar	79.01	$\langle 30, 40, 50, 60, 70\rangle$	565.21	2178.56	188.54	861.111	192.68
8.	bmsWebview1	6.14	$\langle 1, 2, 3, 4, 5, 6\rangle$	106.49	23.70	5.11	111.94	5.12
9.	connect	99.88	$\langle 40, 50, 60, \ldots, 90\rangle$	836.10	3510.22	488.85	1765.89	717.68
10.	bmsWebview2	4.86	$\langle 1, 2, 3, 4\rangle$	70.99	10.23	3.17	75.37	3.16
11.	belgiumRetail	57.48	$\langle 10, 20, 30, 40, 50\rangle$	1444.44	1860.98	48.89	1400.00	64.00
12.	skin	79.25	$\langle 10, 20, 30, \ldots, 70\rangle$	1490.00	3348.70	81.69	2714.29	99.29
13.	accidents	99.99	$\langle 10, 20, 30, \ldots, 90\rangle$	601.27	4526.47	419.98	661.9	585.52
14.	onlineRetail	10.04	$\langle 1, 2, 3, \ldots, 9\rangle$	126.07	56.31	8.69	155.63	8.83
15.	recordLink	99.99	$\langle 10, 20, 30, \ldots, 90\rangle$	1803.25	5200.77	309.71	8583.33	508.19
16.	kosarak	60.75	$\langle 10, 20, 30, \ldots, 60\rangle$	1888.89	2434.53	59.83	4250.00	72.50
17.	kddcup99	79.36	$\langle 10, 20, 30, \ldots, 70\rangle$	1366.71	2893.80	373.32	4000.00	454.31
18.	pamp	94.51	$\langle 10, 20, 30, \ldots, 90\rangle$	658.54	4499.74	425.66	895.82	471.18
19.	uscensus	88.23	$\langle 30, 40, 50, \ldots, 80\rangle$	546.55	2971.37	210.29	890.65	233.58
20.	powerc	96.74	$\langle 10, 20, 30, \ldots, 90\rangle$	1372.14	5406.29	168.90	2125.00	236.17
21.	chainstore	5.73	$\langle 1, 2, 3, 4, 5\rangle$	100.00	16.64	4.00	100.00	4.00

For this, we define the set of points as $\{(s, \lambda_s) \mid \forall s \in S\}$ where $\lambda_s = \frac{1}{|FI(s)|}\sum_{\forall X \in FI(s)} |X|$. Recall the concept of the sequence of *minsup* S is explained above.

In addition, ***positive border average length (PBL)*** and ***negative border average length (NBL)*** are denoted with the same formulation as the previous *FAL* metric. In this case, *PBL* and *NBL* utilize the set $Bd^+(FI(s))$ and $Bd^-(FI(s))$ respectively instead of the $FI(s)$. It is worth mentioning that in this work all the metrics which base their calculation on itemset lengths such as *FAL*, *PBL*, and *NBL* are not normalized.

For their third metric, Palmerini et al. [16] use the concept of entropy over probability distributions of k-itemsets in order to attempt a density estimation of the database without needing to work through it entirely. This is done by first computing H_1 which is the entropy of the 1-itemset set (i.e. the singleton set), then H_2 considering the 2-itemset set, and so forth.

In general they define $H_k = -\sum_{i \in \alpha_k} p_i \log_2(p_i)$, where α_k is the set of k-itemsets, and p_i corresponds to the relative support of a given k-itemset. Datasets with low entropies are considered denser.

We want to point out that most metrics have been described and used in the past, however, we have defined new ones (MCD, GGD) and adapted existing ones to our needs ($ASD, FAL, PBL, PBC, NBC, NBL$).

4 Dataset characterization

In this section we present the characterization of the datasets under study, and introduce our notion of *minimum representative benchmark*. Finally, we include a description of how the *FIM* literature has employed these datasets for algorithm benchmarking.

We consider two public repositories: *FIMI*[2], and *SPMF*[3]; these two repositories have been made available to the community for the purpose of benchmarking new and existing *FIM* algorithms. Both repositories possess collections of real-life datasets (listed in Table 1, from row 3 to row 21). The first two datasets (i.e. row 1 and 2) are also real datasets taken from W. Hamalainen of the University of Eastern Finland[4]. The first four columns of Table 1 present elemental properties of any transactional database: the number of transactions *(DS)*, alphabet size *(AS)*, average transaction size *(ATS)*, and the maximum transaction size *(MTS)*.

[2] `http://fimi.ua.ac.be` (accessed September 1, 2017)

[3] `http://philippe-fournier-viger.com/spmf/` (accessed September 1, 2017)

[4] `http://www.cs.uef.fi/~whamalai/datasets.html` (accessed September 1, 2017)

Table 3: Benchmarking datasets utilized in published empirical studies. Columns with a citation correspond to the studies, and an "x" marks the fact that the dataset has been included in the study. The datasets have been grouped by clusters according to Figure 1.

	Dataset	*Cluster*	*[11]*	*[21]*	*[2]*	*[15]*	*[10]*	*[13]*	*[4]*	*[3]*	*[14]*	*[18]*	*[22]*
4.	foodmart	0											
8.	bmsWebview1	0	x	x	x	x	x					x	x
10.	bmsWebview2	0	x	x	x								x
11.	belgiumRetail	0	x						x				x
14.	onlineRetail	0											
21.	chainstore	0											
12.	skin	1											
16.	kosarak	1	x					x					x
17.	kddcup99	1											
20.	powerc	1											
15.	recordLink	2											
1.	forests	3											
2.	bogPlants	3											
3.	chess	3	x	x	x	x							x
5.	mushroom	3	x	x	x	x	x			x			x
6.	pumsb	3	x	x	x	x			x				x
7.	pumsbStar	3	x	x	x			x					x
9.	connect	3	x	x	x	x		x		x	x		x
13.	accidents	3	x					x	x				x
18.	pamp	3											
19.	uscensus	3											

Metrics *F1* and *GGD* of Table 1 and *MSS* of Table 2 are presented as percentage. Metrics *H1* and *H2* of Table 1 have been calculated based on the formulation of entropy H using 1-itemset and 2-itemset sets, respectively. The five metrics of Table 2—*MCD, ASD, FAL, PBC,* and, *PBL*—are computed at different levels of support. These levels of support are presented in the same table for each dataset and are named by the notation S defined previously in Section 3.

The formulation of the metrics related to negative borders *NBC* and *NBL* have been presented in this work, however, the experimental results on these metrics will be included in a future work.

Two observations are important. Firstly, each of the support levels in S is given in percentage and is denoted as the ratio of the number of transactions to the size of the database. Secondly, that the highest level of support used with a particular dataset is bounded by the datasets' *MSS* value. Given this upper bound, we have taken (roughly) equidistant intervals of support.

Next, we show in Table 1 and Table 2 the values obtained for each elemental metric and also for the more sophisticated metrics described in Section 3 for our 21 datasets.

4.1 Minimum representative benchmark. Intuitively, a representative benchmark is a set of datasets one can safely do empirical studies on (for example, checking whether one algorithm is better than another in a benchmarking study). As an example, suppose that we only care about the number of transactions of a dataset. Then, a benchmark would be representative if it were to include datasets with few transactions as well as datasets with a high number of them, and perhaps datasets with an intermediate number of transactions. In a sense, we are clustering the datasets according to this particular value, and a representative benchmark is one that includes at least a representative from each cluster. We follow this idea, but instead of using a single characteristic, we use a whole array of them – namely, all the metrics considered in Section 3.

Therefore, our working assumption is that if the values of the benchmark cover the whole range of possible values, then it is representative. Moreover, we seek *minimum* benchmarks in the sense that we want to include as few datasets as possible (adding datasets with similar characteristics to the benchmark does not enrich the benchmark and slows down the benchmarking process).

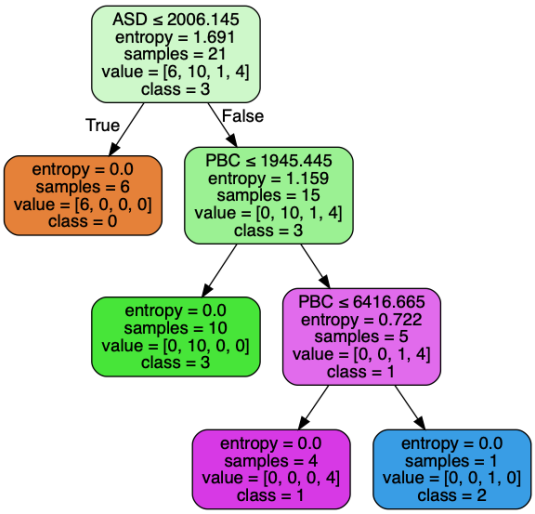

Figure 2: Decision tree classifying datasets into the four clusters found by `K-Means`.

We have clustered the 21 datasets used in this paper into four clusters using `K-Means`. In order to perform the clustering, we have used the `scikit-learn` package [17], more concretely its `kmeans++` initialization version with 500 restarts. Metrics have been scaled prior to clustering with `RobustScaler` from `scikit-learn` to avoid different ranges of values perverting the clustering process. Finally, we have chosen to partition into four clusters because this resulted into the most succint description of the resulting clusters (please see paragraph below on the tree description of the clusters regarding Figure 2).

Figure 1 shows the four clusters found, which are: $\{4, 8, 10, 11, 14, 21\}$ (cluster 0), $\{12, 16, 17, 20\}$ (cluster 1), $\{15\}$ (cluster 2), and finally $\{1, 2, 3, 5, 6, 7, 9, 13, 18, 19\}$ (cluster 3). The underlying idea is that datasets within clusters are similar to each other (in terms of their metrics' values), but different to others in different clusters. Any representative benchmark should therefore include at least one dataset of each of the four clusters found (a representative benchmark constitutes a hitting set for the four clusters).

In order to understand the nature of the four clusters found by `K-Means`, we have generated a decision tree that is able to classify all datasets into their right cluster using metrics' values as attributes. This tree can be seen in Figure 2. Cluster 0, for example, is formed by datasets having a value of at most 2006 in the ASD metric. Cluster 1 consists of those datasets having large value in ASD and a value for PBC between 1945 and 6416. Cluster 3 consists of datasets having large values in ASD but small values for PBC. Finally, cluster 2 consists of datasets having large values for ASD and PBC.

4.2 Literature review of empirical studies. Table 3 presents the real-life benchmarking datasets used by several authors to performing comparison studies on *FIM* algorithms. In here, Goethals and Zaki [11] and Uno et al. [22] based their benchmarking studies on most of the datasets presented in this work and closely followed by Uno et al. [21] and Burdick et al. [2]. This information allows us to identify which works need to expand their pool of benchmarking datasets in order to consider their outcome as closer to reality, in addition to giving the *FIM* community a global vision of the way the set of public benchmarking datasets is distributed among the different authors.

In order to find out which of the sets employed in the empirical studies of Table 3 are *representative*, we need to establish which of the benchmarks employed constitute hitting sets of the four groups defined previously in Section 4.1.

The datasets used in Goethals and Zaki [11] and Uno et al. [22] are $\{3, 5, 6, 7, 8, 9, 10, 11, 13, 16\}$. This set intersects with all the clusters with the exception of `recordLink` dataset (cluster 2) thus they miss the characteristic metrics provided by this cluster. So, both studies would only need to include `recordLink` to comply with the requirement. The remaining studies mentioned in Table 3 miss datasets from more than one cluster.

Hence, according to the analysis and criteria used in this work, we conclude that there is no study that uses a representative benchmark which is very useful for the various authors to take into account when deciding the set of benchmarking datasets to be used in their experiments. Besides, this work serves as a guide and motivation for new benchmarking datasets to be introduced into the public domain in order to be further characterized and extend the range of characteristics coverage.

5 Conclusions and future work

In this work we provide a study of metrics for the characterization of transactional databases commonly used for the benchmarking of itemset mining algorithms. We argue that these metrics are needed in order to assess the diversity and, therefore, the representativeness of such benchmarks so that conclusions drawn from benchmarking studies are sufficiently supported. We study the representativeness of databases used as the basis for existing benchmarking studies found in the literature based on our characteristics, thus assessing which

of the studies has a better support for their claims. Finally, we propose a way of obtaining benchmarks with guaranteed diversity that authors of new benchmarking studies can benefit from.

As future lines of research we are always on the lookout for new metrics used for the characterization of databases. As new metrics and databases are incorporated into our lists, we should revise the *minimum representative benchmarks* accordingly. This could be thought of as an evolving process and as such could (and should) potentially be included in the *FIMI* or similar repository to be used by the data mining community.

Finally, a line of research that we are pursuing is that of synthetic database generation. The existence of this paper is, in fact, due to the fact that we identified the need for characterizing databases so that we could assess in some ways the datasets that we generate. With characterization metrics, we can check the nature of the databases that we generate (generally mimicking some original database that due to privacy and ownership constraints cannot be shown or used). For example, we could check whether synthetic and original are similar enough, if that is what is desired. Or we could study how parameter tuning in the generative algorithms affects the nature of the generated databases. In any case, being able to characterize databases is a powerful and necessary tool to understand the nature of the generated datasets.

Acknowledgements

We thank the anonymous reviewers for their helpful comments. Both authors have been partially supported by TIN2017-89244-R from MINECO (Spain's Ministerio de Economia, Industria y Competitividad) and the recognition 2017SGR-856 (MACDA) from AGAUR (Generalitat de Catalunya). Christian Lezcano is supported by Paraguay's Foreign Postgraduate Scholarship Programme Don Carlos Antonio López (BECAL).

References

[1] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc. ISBN 1-55860-153-8.

[2] Douglas Burdick, Manuel Calimlim, Jason Flannick, Johannes Gehrke, and Tomi Yiu. MAFIA: A performance study of mining maximal frequent itemsets. In *Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations*, 2003.

[3] Zhi-Hong Deng and Sheng-Long Lv. Fast mining frequent itemsets using nodesets. *Expert Systems with Applications*, 41(10):4505–4512, 2014. ISSN 0957-4174.

[4] ZhiHong Deng, ZhongHui Wang, and JiaJian Jiang. A new algorithm for fast mining frequent itemsets using N-lists. *Science China Information Sciences*, 55(9):2008–2030, Sep 2012. ISSN 1869-1919.

[5] Nicolas Durand and Mohamed Quafafou. Frequent Itemset Border Approximation by Dualization. *Transactions on Large-Scale Data- and Knowledge-Centered Systems (TLDKS)*, 26:32–60, March 2016.

[6] Frederic Flouvat, F De March, and Jean-Marc Petit. A thorough experimental study of datasets for frequent itemsets. In *Data Mining, Fifth IEEE International Conference on*, pages 8–pp. IEEE, 2005.

[7] Frédéric Flouvat, Fabien De Marchi, and Jean-Marc Petit. A new classification of datasets for frequent itemsets. *Journal of Intelligent Information Systems*, 34(1):1–19, Feb 2010. ISSN 1573-7675.

[8] Philippe Fournier-Viger, Jerry Chun-Wei Lin, Antonio Gomariz, Ted Gueniche, Azadeh Soltani, Zhihong Deng, and Hoang Thanh Lam. The SPMF open-source data mining library version 2. In Bettina Berendt, Björn Bringmann, Élisa Fromont, Gemma Garriga, Pauli Miettinen, Nikolaj Tatti, and Volker Tresp, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 36–40, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46131-1.

[9] Philippe Fournier-Viger, Jerry Chun-Wei Lin, Bay Vo, Tin Chi Truong, Ji Zhang, and Hoai Bac Le. A survey of itemset mining. *Wiley Interdisc. Rew.: Data Mining and Knowledge Discovery*, 7(4), 2017.

[10] Bart Goethals. Survey on frequent pattern mining. Technical report, University of Helsinki, 2003.

[11] Bart Goethals and Mohammed Javeed Zaki. Advances in frequent itemset mining implementations: Introduction to FIMI03. In *FIMI '03, Frequent Itemset Mining Implementations, Proceedings of the ICDM 2003 Workshop on Frequent Itemset Mining Implementations, 19 December 2003, Melbourne, Florida, USA*, 2003. URL `http://ceur-ws.org/Vol-90/intro.pdf`.

[12] Karam Gouda and Mohammed Javeed Zaki. Efficiently mining maximal frequent itemsets. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, ICDM '01, pages 163–170, Washington, DC, USA, 2001. IEEE Computer Society. ISBN 0-7695-1119-8.

[13] G. Grahne and J. Zhu. Fast algorithms for frequent itemset mining using FP-trees. *IEEE Transactions on Knowledge and Data Engineering*, 17(10):1347–1362, Oct 2005. ISSN 1041-4347.

[14] Jiawei Han, Jian Pei, Yiwen Yin, and Runying Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Mining and Knowledge Discovery*, 8(1):53–87, Jan 2004. ISSN 1573-756X.

[15] Guimei Liu, Hongjun Lu, and Jeffrey Xu Yu. AFOPT: An efficient implementation of pattern growth approach. In *In Proceedings of the ICDM workshop*, 2003.

[16] P. Palmerini, S. Orlando, and R. Perego. Statistical properties of transactional databases. In *Proceedings of the 2004 ACM Symposium on Applied Computing*, SAC '04, pages 515–519, New York, NY, USA, 2004. ACM. ISBN 1-58113-812-1.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python . *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[18] Jian Pei, Jiawei Han, Hongjun Lu, Shojiro Nishio, Shiwei Tang, and Dongqing Yang. H-mine: hyper-structure mining of frequent patterns in large databases. In *Proceedings 2001 IEEE International Conference on Data Mining*, pages 441–448, 2001.

[19] Ganesh Ramesh, William A. Maniatty, and Mohammed J. Zaki. Feasible itemset distributions in data mining: Theory and application. In *Proceedings of the Twenty-second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '03, pages 284–295, New York, NY, USA, 2003. ACM. ISBN 1-58113-670-6.

[20] Ganesh Ramesh, Mohammed J. Zaki, and William A. Maniatty. Distribution-based synthetic database generation techniques for itemset mining. In *Proceedings of the 9th International Database Engineering & Application Symposium*, IDEAS '05, pages 307–316, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2404-4.

[21] Takeaki Uno, Tatsuya Asai, Yuzo Uchida, and Hiroki Arimura. LCM: An efficient algorithm for enumerating frequent closed item sets. In *FIMI*, 2003.

[22] Takeaki Uno, Masashi Kiyomi, and Hiroki Arimura. LCM ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets. In *FIMI*, 2004.

[23] Mohammed J. Zaki. Scalable algorithms for association mining. *IEEE Trans. on Knowl. and Data Eng.*, 12(3):372–390, May 2000. ISSN 1041-4347. doi: 10.1109/69.846291.

[24] Zijian Zheng, Ron Kohavi, and Llew Mason. Real world performance of association rule algorithms. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 401–406, New York, NY, USA, 2001. ACM. ISBN 1-58113-391-X.

[25] Albrecht Zimmermann. The data problem in data mining. *ACM SIGKDD Explorations Newsletter*, 16(2):38–45, 2015.