Frequent Itemset Mining: a 25 Years Review

José María Luna[*], Philippe Fournier-Viger[†], Sebastián Ventura[‡][§]

Article Type:

Advanced Review

Abstract

Frequent itemset mining is an essential task within data analysis since it is responsible for extracting frequently occurring events, patterns or items in data. Insights from such pattern analysis offer important benefits in decision making processes. However, algorithmic solutions for mining such kind of patterns are not straightforward since the computational complexity exponentially increases with the number of items in data. This issue, together with the significant memory consumption that is needed in the mining process, make it necessary to propose extremely efficient solutions. Since the frequent itemset mining problem was first described in the early nineties, multiple solutions have been proposed by considering centralized systems as well as parallel (shared or non shared memory) architectures. Solutions can also be divided into exhaustive search and non-exhaustive search models. Many of such approaches are extensions of other solutions and it is therefore necessary to analyse how this task has been considered during the last decades.

[*]Department of Computer Science and Numerical Analysis, University of Cordoba, Spain

[†]School of Humanities and Social Sciences, Harbin Institute of Technology (Shenzhen), Shenzhen, Guangdong, China, 518055

[‡]Department of Computer Science and Numerical Analysis, University of Cordoba, Spain

[§]Faculty of Computing and Information Technology, King Abdulaziz University, Saudi Arabia

INTRODUCTION

To live in the Big Data Era implies data being gathered everywhere, at every moment, from different devices and, most of the time, in an almost imperceptible way. Taking advantage of such information is essential for many organizations as well as governments in decision-making to improve our daily life (Kraska, 2013). Data analytic systems are therefore booming thanks to their capacity to extract hidden, effective and usable knowledge from large collections of data. Though many different tasks come under the umbrella of data analysis or data mining, Frequent Itemset Mining (FIM) is, from the very outset, an essential task due to its ability to extract frequently occurring events, patterns or items (symbols or values) in data (Aggarwal & Han, 2014).

In the process of transforming raw data into significant and meaningful information for making sense of the data, the key element is the pattern (a singleton or set of items) which represents any type of homogeneity and regularity, and it is therefore considered as a good descriptor of intrinsic and important properties of the data (Han & Kamber, 2000). Numerous FIM algorithms have been proposed since the first approach was described at the beginning of the nineties (Agrawal, Imielinski, & Swami, 1993). In that approach, a levelwise breadth first search methodology was responsible for producing candidate itemsets whose frequency counting was performed by reading the dataset multiple times (one for each size of candidate itemsets). Later algorithms such as FP-Growth (Han, Pei, & Yin, 2000) and Eclat (Zaki, 2000), on the contrary, were based on a depth-first search procedure considering a prefix-tree-based main memory compressed representation of the input dataset or a vertical transposition of the dataset to work on transaction identifiers, respectively. To date, really efficient algorithms can be found in literature for solving many issues related to FIM (memory requirements, exponential time complexity, data dimensionality, search space pruning, etc) and, in many situations, such algorithms are suitable enough for facing the problems (Aggarwal & Han, 2014). However, as a consequence of the large volumes of information to be analysed, such traditional (centralized) FIM approaches have laid the foundations for novel methodologies based on parallel programming architectures considering both shared and distributed (non shared) memory (Moens, Aksehirli, & Goethals, 2013).

Since 1993, when the first FIM algorithm was released (Agrawal et al., 1993), a special attention has been given to the performance of novel algorithms in this field (Borgelt, 2012; Fournier-Viger et al., 2017). Nowadays, 25 years later, extremely large datasets can be analysed in few seconds and this is not only a matter of novel architectures and hardware progresses but also a consequence of the proposed algorithmic solutions. In this regard, even when multiple articles have been already published as comparative studies concerning parallel FIM approaches on modern architectures (Apiletti et al., 2017), it is interesting to describe the timeline of FIM during the last 25 years considering both centralized and parallel models as well as any methodology (exhaustive or heuristic) proposed so far. This descriptive analysis is essential to be aware of how the FIM task has improved over time and why recent approaches are based on others that have been proposed a decade ago. It is noteworthy that the mere fact of understanding how/why a recent algorithm reuses or adapts some algorithmic concepts of previous approaches might be the key fact for new ideas flowing.

This paper is organized as follows. The first section presents general information about FIM, its key applications as well as some related tasks. Then a summary of how FIM proposals have improved during the last 25 years is presented. The most important approaches (sequential, multi-thread and distributed) are then described in subsequent sections. Finally, some lessons learned are outlined.

FREQUENT ITEMSET MINING

FIM is the task of extracting any existing frequent itemset (having an occurrence frequency no less than some threshold) in data. This task was proposed in the early nineties for discovering frequently co-occurring items in market basket analysis (Agrawal et al., 1993), and was initially called *large itemset mining.* Some formal definitions related to this task, different application domains as well as multiple related tasks are described in this section.

Formal Definitions

Definition 1 (Pattern). A pattern is defined as a set of elements, events or items that regularly co-occur in a database (Aggarwal & Han, 2014). Formally speaking, a pattern P in a database Ω is defined as a subset of items $P \subseteq \{i_1, ..., i_n\} \in \Omega$ that describes valuable features of data.

Definition 2 (Frequency of a pattern). The frequency of a pattern P is denoted as the number of data records, or percentage of them, that include the subset of items defined by P. In other words, given a set of items $I = \{i_1, ..., i_n\}$, a database Ω comprising $R = \{r_1, r_2, ..., r_m\} \in \Omega$ data records, and each record including a set of items $\forall r \in R : r \subseteq I$, the frequency of P is denoted as $|\{\forall r \in R : P \subseteq r, r \subseteq I \in \Omega\}|$.

The FIM task is a really tough problem due to the large number of candidate patterns (itemsets) that are required to be handled. Considering a dataset comprising n items the number of patterns (or itemsets) of size k is $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ for any $k \leq n$. As a result, the total number of feasible patterns is $2^n - 1$ for such dataset, and the complexity of finding patterns of interest is in exponential order. This complexity is even higher when the frequency of each pattern is calculated, resulting as $O((2^n - 1) \times m \times n)$ for a dataset comprising m records and n items. Aiming at shedding some light on the complexity, a dataset comprising just 33 items (a low number) produces more than 8.5×10^9 feasible patterns, a number that is higher than the world population in 2018 (estimated as 7.7 billion people).

Generally, an overwhelming number of solutions is impractical for any user (data scientist, company, goverment, etc) who wants to get actionable insights. In this regard, interestingness quality measures (Luna, Ondra, Fardoun, & Ventura, 2018) can be used to filter and/or rank the output. These measures are divided into *objective* or data-driven (statistical and structural properties of data) and *subjective* or user-driven (user's preferences and goals). From the FIM definition, a filtering of the search space and, hence, the resulting set of solutions was considered through the anti-monotone property of support —a.k.a frequency (see Definition 2). It is, actually, the keystone of FIM and most of existing quality measures (either objective or subjective) are based on such metric (Tan, Kumar, & Srivastava, 2004).

Definition 3 (Anti-monotone). A constraint c is anti-monotone if an element violates

Symbol	Meaning
P	Pattern
Ω	Dataset
R	Set of records included in a dataset
r	Single record
I	Set of items included in a dataset
i	Single item
m	Number of different records included in data
n	Number of different items included in data
k	Number of items included in a pattern

Table 1: Lookup table with some important symbols and their meanings.

constraint c, so does any of its supersets (Aggarwal & Han, 2014). Thus, if an itemset does not satisfy the minimum frequency threshold, then all of its supersets must also be infrequent. Conversely, if an itemset is frequent, then all of its subsets must also be frequent.

Finally, a summary of the symbols and their meanings is provided in Table 1.

Related Tasks

The FIM problem is considered as the root of the pattern mining field, which encompasses multiple tasks that aim at extracting itemsets on multiple forms and for various purposes (Aggarwal & Han, 2014). The simplest variation of FIM is the extraction of itemsets that do not frequently appear in data or those that were discarded by the anti-monotone property of support in FIM. It gave rise to the definition of pattern mining as the problem of mining sets of items that frequently (or infrequently) appear in data (Koh & Ravana, 2016). Starting from it, the pattern mining concept was extended with new ideas (Aggarwal & Han, 2014), considering small variations in some cases, like the one considering an item within a pattern (either frequent or infrequent) as interesting not only if it is present (postive patterns) but also if it is absent (negative patterns) in data (Savasere, Omiecinski, & Navathe, 1998; H. Wang, Zhang, & Chen, 2008). Of course, an item and its opposite form (positive or negative) always produce a zero support value since any record cannot satisfy both at the same time.

Nowadays, with the increasing interest in data storage, patterns are required to be extracted not only from tabular and static data but from various advanced data types. An example is streaming data where sequences of data records are arriving at different periods of time (Gama, 2010). In some other situations, gathered data is arranged sequentially so the aim is to seek patterns (sequences of events) that frequently appear in a collection of data records (Fournier-Viger, Lin, Kiran, & Koh, 2017). Sometimes, sequential data also include features related to the timestamp and/or location (trajectory of a moving object) (Zheng, 2015). In some other application fields (bioinformatics, cheminformatics, social network) data is better represented using graphs than using a flat representation. The aim is to obtain frequent graph patterns that are useful to characterize, discriminate or cluster groups of graphs (Aggarwal & Han, 2014; Jiang, Coenen, & Zito, 2013). Additionally, recent studies are paying attention to patterns, for example in market basket analysis, that generate a high profit (Fournier-Viger, Chun-Wei Lin, Vo, & Tseng, 2019), or low profit even though they are frequently purchased (Gan et al., 2018). In other words, high utility data consider that all items are not always equally important. Similarly, weighted frequent patterns (Cai, Fu, Cheng, & Kwong, 1998) were proposed to denote that not all the items are equally important in data and, therefore, it is required to provide some weights according to their significance. In some other situations, uncertainty may be present in data due to different reasons so users may not be certain about the presence or absence of an item or record and this can be approached by probabilistic proposals (Aggarwal & Han, 2014) as well as possibilistic solutions (Guil, Juárez, & Marín, 2007). Finally, the use of small threshold values might overwhelm the user with many redundant frequent itemsets so multiples solutions have been proposed to extract compact representations of multiple frequent itemsets (Aggarwal & Han, 2014) or general representations through generalized itemsets (Srikant & Agrawal, 1995).

FIM is generally related to descriptive tasks, which aim at finding comprehensible patterns that denote any interesting behaviour on unlabeled data. The prime descriptive task associated to FIM is association rule mining (Zhang & Zhang, 2002), which was simultaneously proposed in the early nineties. An association rule describes correlations among items within a frequent itemset, defining an implication of the form $X \rightarrow Y$ where both X (antecedent) and Y (consequent) constitute the itemset $X \cup Y$. The meaning of an association rule is that if the antecedent X is satisfied, then it is highly probable that the consequent Y is also satisfied. Almost from the beginning (Agrawal et al., 1993), both association rule mining and frequent itemset mining have been used interchangeably since when one finds a frequent association rule he also obtain a frequent pattern. Currently, modern studies are considering descriptive tasks to understand an underlying phenomena (according to a target variable), giving rise to the concept of supervised descriptive pattern mining (Ventura & Luna, 2018). This concept gathers multiple tasks including contrast set mining, emerging pattern mining, subgroup discovery, class association rules, and exceptional model mining.

Key Applications of FIM

The spark application for FIM was market basket analysis, in which customers' behavior was gathered as baskets of items bought together (Agrawal et al., 1993). A rather old, but still significant, example of a frequent pattern is the $\{Beer, Diapers\}$ itemset, which suggests that both items tend to be bought at the same time. Insights like the previous one are useful to launch the right advertising campaign (according to customers' buying habits) or to place products accordingly on the shelves to increase sales (Aggarwal & Han, 2014).

FIM has broaden its horizons and, nowadays, it is being applied to fields such as text mining to both discover word co-occurrences in terms of adjacency and finding frequent bigrams, trigrams, or phrases in texts (Aggarwal & Yu, 2001). The analysis of whether the frequency of phrases within a text increase or decrease over time is essential to provide valuable hints about trends (social networks or news). This specific analysis is, for example, a key issue in medical domains to identify and understand the spread of infectious diseases (Ventura & Luna, 2018). FIM has also been considered for describing the risk of heart diseases, easing the medical practitioner labour to make diagnostic decisions and determine the risk level of patients at an early stage (Gamberger & Lavrac, 2002). Changing to the biological field, FIM has been considered to identify common and useful properties of the underlying data (sequences of complex biological molecules, microarrays or protein interaction networks) for a variety of medical purposes (genes related to colon cancer, gene expression profiles of subtypes of acute lymphoblastic leukemia, gene expressions for breast tumor, etc) (J. Li & Wong, 2002).

The previous are just a few examples of FIM applications, but plenty of real-world problems have been already addressed by this task. To list a few, FIM has been correctly applied to education by analysing data gathered by e-learning systems with the aim of providing instructors with beneficial or detrimental relationships between the use of educational resources and the student's learning (Romero, Zafra, Luna, & Ventura, 2013). FIM has also been used as a recommender system to support the students' final degree decision according to their skills and preferences during the high school period (Noaman, Luna, Ragab, & Ventura, 2016). In crime against women, the well-known Apriori algorithm has been correctly apply (Bansal & Bhambhu, 2014). FIM has also been applied to chemical domains to find relevant substructures of specific molecules (Dehaspe, Toivonen, & King, 1998). As a last example, FIM has been used in sociology (Ruggieri, Pedreschi, & Turini, 2010) to detect different social discrimination (gender, race, sexuality, etc).

25 YEARS OF IMPROVEMENTS

FIM was first proposed in 1993 (Agrawal et al., 1993) in the context of market basket analysis and, since then, many improvements have been proposed (see Figure 1). In this section, the crucial events that have occurred over the last 25 years are outlined to understand how and what advances have been reached in a general sense. Thus, this section only includes what the authors consider as key algorithms for the improvement of the FIM field —these and additional algorithms are described in the following sections.

The first FIM proposal, named Apriori (Agrawal et al., 1993), was designed to work on a set of binary attributes (called items) and a database of transactions where each transaction was represented as a binary vector. To seek frequent itemsets, this proposal followed a levelwise breadth first search methodology where, in each level, candidate itemsets were formed from the already mined (frequent) itemsets. Two years later, different authors proposed an improvement of FIM that reduced the number of disk I/Os (Savasere, Omiecinski, & Navathe, 1995). With this aim, the algorithm divided the dataset into a number of non-overlapping partitions such that each partition could be accommodated in main memory and, therefore, partitions were read only once during each phase. Frequent local itemsets

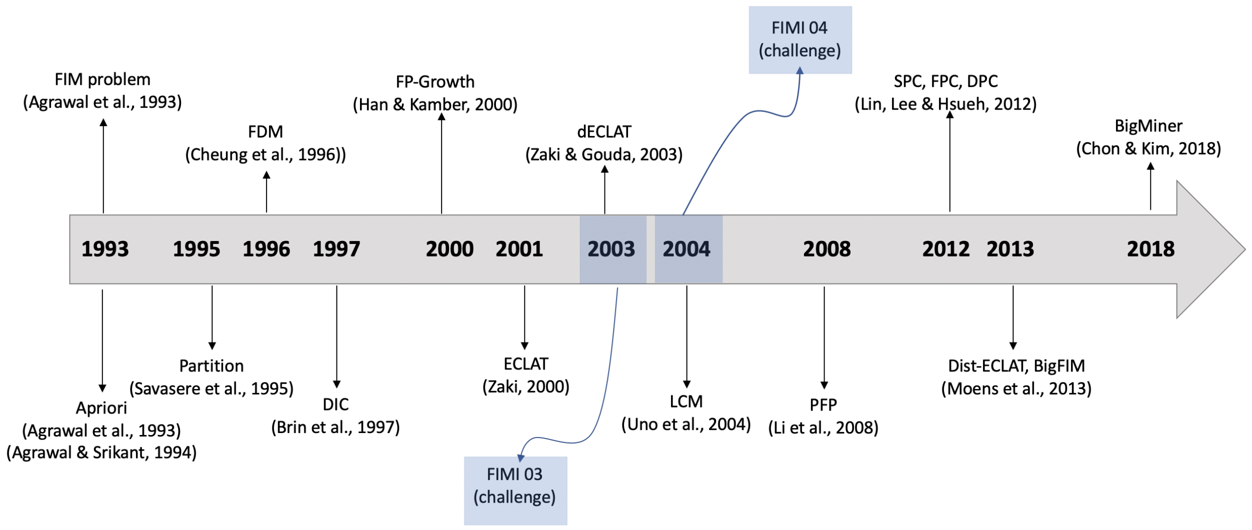

Figure 1: Timeline of FIM through 25 years.

were extracted from each partition whereas potential frequent global itemsets were obtained in a second stage by considering the local frequent itemsets previously mined. A year later, in 1996, mining patterns in a distributed environment was proposed (Cheung, Han, Ng, Fu, & Fu, 1996). In this approach, authors also considered the concept of local and global patterns to propose a wide number of pruning strategies.

Considering just the three previous studies it is demonstrated that, only three years after the first FIM algorithm was proposed, researchers were enterely aware of the tough problem FIM was and the importance of proposing not only efficient algorithmic solutions but also new approaches (parallel and distributed computing) to handle such problem. From that moment on, the performance of FIM has been improved in many different ways. DIC (Dynamic Itemset Counting) was the first algorithm to improve on Apriori, but that improvement was quite small and it performed even worse in some situations (Brin, Motwani, Ullman, & Tsur, 1997). It was in 2000 that one of the most important algorithmic improvement was proposed using a novel frequent pattern tree structure for storing compressed information about patterns. This algorithm, named FP-Growth (Han et al., 2000), was about an order of magnitude faster than Apriori and, nowadays, it is considered as a key algorithm to compare with. Almost at the same time, in 2001, the ECLAT (Equivalent CLAss Transformation) algorithm was proposed associates to each pattern the list of transactions in which it occurs

(Zaki, 2000). The frequency of two patterns can be therefore simply obtained by intersecting their lists. However, the efficiency of this method decreases when a high number of frequent patterns (extremely large lists of transactions) needs to be intersected. With this aim, a similar algorithm based on the diffset data structure (dECLAT) was proposed right afterwards, which only keeps track of differences in the transactions of a candidate pattern from its generating frequent patterns (Zaki & Gouda, 2003). Thanks to this improvement, dECLAT was reported to outperform by several orders of magnitude the running time of ECLAT and FP-Growth on dense datasets (when items occur in most transactions). Two important competitions took place in subsequent years under the name Frequent Itemset Mining Implemenations (FIMI'03 and FIMI'04) in which plenty of implementations and solutions were proposed. One of the most important algorithms to date, named LCM (Uno, Kiyomi, & Arimura, 2004), was the winner of the FIMI'04 competition. The number of proposed solutions was so high and the performance of the solutions were so good that, nowadays, few faster algorithms from a sequential point of view have be found and, therefore, many authors have considered the problem as solved. It was around these years that some authors decided to address the FIM problem using an heuristic search (Mata, Alvarez, & Riquelme, 2001) instead of an exhaustive one as traditional approaches in this field do. From that year on, different algorithms have been designed using multiple non-exhaustive methodologies (Ventura & Luna, 2016).

The term Big Data was first used around 2005 and it transformed all the data analytics subfields, including FIM. In the next few years, the MapReduce programming framework was introduced to industry enabling to process data in parallel as small chunks in distributed clusters (Dean & Ghemawat, 2008). The first FIM proposal based on MapReduce came promptly and it was a version of the FP-Growth algorithm, called PFP (H. Li, Wang, Zhang, Zhang, & Chang, 2008), to work on distributed machines. After that, some authors have designed initial adaptations (SPC, FPC, DPC) of Apriori on MapReduce (Lin, Lee, & Hsueh, 2012). However, it was not till 2008 when truly efficient MapReduce algorithms for FIM were proposed (DistECLAT and BigFIM) (Moens et al., 2013), one based on the well-known ECLAT algorithm and the other combining principles from both Apriori and Eclat. Recently, novel MapReduce methods for FIM have been proposed (Chon & Kim, 2018; Luna, Padillo, Pechenizkiy, & Ventura, 2018) to alleviate problems associated to the MapReduce framework, that is, workload skewness, amount of intermediate data, and network communication overhead. Nevertheless, even though all these MapReduce proposals seems to be *new*, the distributed methodology is based on the solution provided in 1995 in which the dataset is split into subsets for subsequent analysis (Savasere et al., 1995). Thus, it is important to note that the key improvements carried out thanks to MapReduce are more derived from the novelty of the architecture than being major algorithmic breakthroughs.

SEQUENTIAL SOLUTIONS

The FIM problem has been sequentially solved through two general perspectives, that is, exhaustive search (Aggarwal & Han, 2014) and heuristic-based (non-exhaustive) (Ventura & Luna, 2016) methodologies. Which of these two methodologies is better is not a trivial issue and it depends on multiple factors. Exhaustive search methods are able to extract any existing pattern in data, which is a requirement in many fields. Potentially missing some patterns is a real handicap for non-exhaustive search methodologies since they do not guarantee that important solutions will not be ignored by the mining process. But on the contrary, guiding the mining process by an heuristic search widely reduces the runtime required, which is a crucial drawback of exhaustive-search solutions. Even though many solutions were proposed to improve the performance of sequential solutions, they are generally focused on either exhaustive or heuristic-based algorithmic solutions, where some authors achieved excellent runtimes by proposing novel data structures (Luna, Cano, Pechenizkiy, & Ventura, 2016). The idea is to organize and restructure the input data so that data access becomes faster without doing major changes to the algorithms.

In general terms, really interesting algorithmic solutions (Aggarwal & Han, 2014; Ventura & Luna, 2016) have been proposed since the proposal of the first exhaustive search algorithm in 1993 (Agrawal et al., 1993). This algorithm, known as Apriori, is based on the concept of extension defined as follows. For an itemset X formed from a list of items in data, then $X \cup Y$ is an extension of X if $X \cap Y = \emptyset$. Apriori was presented as a levelwise breadth first search methodology where, in each level, itemsets are extended and their frequency are measured in terms of number of transactions in which each itemset appears. Each extended itemset is then added to the new set (corresponding to the specific level) if its frequency surpassed a predefined threshold. This process is repeated till no new itemsets can be formed. Despite that Apriori is the most-well known algorithm, it is by far the one having the worse performance. Together with Apriori, the same authors proposed two additional algorithms (Agrawal & Srikant, 1994) named AprioriTid and AprioriHybrid. The AprioriTid algorithm achieves a better performance for frequency counting by replacing each transaction in the database by the set of candidate itemsets that occur in that transaction. This algorithm, however, performed much slower than Apriori in early iterations. In this regard, AprioriHybrid was proposed as an improvement of AprioriTid that applies Apriori for the initial iterations and switches to AprioriTid for the subsequent iterations. Nevertheless not so huge performance improvements were achieved. Almost at the same time, other authors proposed the Partition algorithm (Savasere et al., 1995) with the aim of improving the performance of FIM by reducing the number of disk I/Os. The idea was to split data into chunks so that they are kept into main memory and local frequent itemsets were obtained from each data subset. It is important to highlight that this methodology has been the basis for the ensuing distributed computing solutions (most of them presented over a decade later). In that year, some authors proposed the DHP (Direct Hashing and Pruning) algorithm (Park, Chen, & Yu, 1995) that introduced a hashing methodology to improve the performance of existing approaches.

At this point in the development of FIM it was known that main factors governed the performance: the number of passes made over all the data (this number is related to the maximum number of elements in a candidate itemset) and the efficiency of those passes. In this regard, DIC (Dynamic Itemset Counting) was designed to reduce the number of passes by considering a number of intervals including M transactions (Brin et al., 1997). It counts the occurrence frequency of each itemset in these ranges and frequent itemsets in each range will be used to produce a superitemset. This means that it is not required to wait till the end of the current pass to start counting the frequency of an itemset. However, the main drawback of this approach is the value of M itself since small M values imply starting counting itemsets very early but, meanwhile, it produces a considerable overhead.

Any FIM algorithm proposed till 2000 was a small variation of Apriori and it was in this year when one of the most important improvements was achieved by means of a novel frequent pattern tree structure for storing compressed information about patterns (Han et al., 2000). Thanks to this algorithm, named FP-Growth, if multiple transactions share an identical frequent itemset, then they can be merged into one. The database is therefore read once and frequent itemsets are obtained by traversing the resulting tree structure. Nowadays, FP-Growth is a key algorithm to compare with since it runs about an order of magnitude faster than Apriori. However, when the database is huge and sparse, the frequent pattern tree structure considered by FP-Growth will be large and the space requirement for recursion is a challenge. In this sense, the same authors of FP-Growth proposed a novel data structure and mining method, H-Mine (Pei et al., 2001). Almost at the same time, different authors proposed the ECLAT (Equivalent CLAss Transformation) algorithm in which each itemset was associated with the list of transactions in which it appears (Zaki, 2000). The frequency of an itemset is calculated as the number of elements in the list. Two itemsets can be merged simply by intersecting their lists. ECLAT achieved a truly good performance (better than FP-Growth in some cases) but this method is not appealing for dealing with a high number of frequent patterns that require to intersect an extremely large lists of transactions. A great improvement in runtime (several orders of magnitude with regard to ECLAT and two times with respect to FP-Growth) was achieved right afterwards thanks to dECLAT (Zaki & Gouda, 2003). This algorithm keeps track of differences in the transactions where a candidate pattern appears compared to its generating frequent patterns. Results were so favorable that, nowadays, this is considered as one of the fastest sequential algorithm. At about the same time, the LCM (Uno et al., 2004) algorithm was proposed, which performs a depth-first search and adopts two key techniques to reduce the cost of data scans: (1) creating projections of databases for each pattern, and (2) merging identical transactions in these databases. This algorithm was the winner of the FIMI'04 competition. Then, two other interesting FIM solutions were proposed, that is, RElim (Recursive Elimination) (Borgelt, 2005) and SaM (Split and Merge) (Borgelt, 2010). In both cases, an initial data scan is done to calculate the frequencies of items and infrequent ones are discarded from the transactions. In each transaction, items are ordered according to their frequency and transactions are then separated according to their leading item (the most frequent one), thus resulting in a vertical representation. The only difference between RElim and SaM is the way in which transactions are grouped according to their leading item. Multiple studies, however, have demonstrated that FP-Growth, ECLAT and dECLAT outperform RElim and SaM in a wide range of situations.

In 2001, at a time where the FIM problem was mainly solved through exhaustive search, the first non-exhaustive search algorithm was proposed (Mata et al., 2001). The main advantage of this methodology is the mere fact that the runtime is almost constant for whatever dataset is analysed, but a drawback is the lack of guarantee that the whole search space is analyzed (Ventura & Luna, 2016). Aside from the runtime, evolutionary computation solutions in FIM gained interest among researchers due to their ability to cope with continuous features without performing a discretization step — impossible for exhaustive search solutions. Thus, most of the existing solutions in this field are based on such feature, optimizing the range of values. Some representative algorithms are Alatas (Alatas & Akin, 2006), QuantMiner (Salleb-Aouissi, Vrain, & Nortet, 2007), G3PARM (Luna, Romero, & Ventura, 2012), and G3P-Quantitative (Luna, Romero, Romero, & Ventura, 2014). Although multiple non-exhaustive solutions have been proposed so far, it is EARMGA (Yan, Zhang, & Zhang, 2009) that has obtained the most citations, mainly because it was the first one that did not require using a fixed minimum support threshold.

As a summary of sequential solutions for FIM, Table 2 shows the most well-known algorithms among the research community. Apriori, FP-Growth, ECLAT and dECLAT appear in most of the comparative studies in FIM and they are, by far, the key approaches in the field. Apriori was indeed presented in two different articles (Agrawal et al., 1993; Agrawal & Srikant, 1994) and thus when adding the citations, the total is more than 45,000. The AprioriTid and AprioriHybrid seem to be well-known due to the large numbers of citations. However, this is simply because they were proposed in one of the article presenting Apriori. The Partition and DIC algorithms share ideas with trending solutions for distributed computing (Partition) and data stream mining (DIC). Hence, this explains the huge number of citations for these two approaches —recent methodologies include references to them. As a result, and according to the number of citations, it is demonstrated that exhaustive search

Algorithm	Type	Reference	Year	#Citations GoogleScholar
Apriori	Exhaustive	(Agrawal et al., 1993)	1993	21,056
Apriori	Exhaustive	(Agrawal & Srikant, 1994)	1994	24,230
AprioriTid	Exhaustive	(Agrawal & Srikant, 1994)	1994	24,230
AprioriHybrid	Exhaustive	(Agrawal & Srikant, 1994)	1994	24,230
Partition	Exhaustive	(Savasere et al., 1995)	1995	2,621
DHP	Exhaustive	(Park et al., 1995)	1995	2,329
DIC	Exhaustive	(Brin et al., 1997)	1997	2,672
FP-Growth	Exhaustive	(Han et al., 2000)	2000	8,192
ECLAT	Exhaustive	(Zaki, 2000)	2001	1,603
H-Mine	Exhaustive	(Pei et al., 2001)	2001	511
GENAR	Non-Exhaustive	(Mata et al., 2001)	2001	61
dECLAT	Exhaustive	(Zaki & Gouda, 2003)	2003	704
LCM	Exhaustive	(Uno et al., 2004)	2004	443
RElim	Exhaustive	(Borgelt, 2005)	2005	87
Alatas	Non-Exhaustive	(Alatas & Akin, 2006)	2006	94
QuantMiner	Non-Exhaustive	(Salleb-Aouissi et al., 2007)	2007	128
EARMGA	Non-Exhaustive	(Yan et al., 2009)	2009	194
SaM	Exhaustive	(Borgelt, 2010)	2010	39
G3PARM	Non-Exhaustive	(Luna et al., 2012)	2012	61
G3P-Quantitative	Non-Exhaustive	(Luna et al., 2014)	2014	37

Table 2: List of the most important sequential FIM solutions ordered by year. Algorithms highlighted in gray denote key solutions in the field (those used in most of recent comparative analyses). Last updated May 20th, 2019.

methods are desirable for the FIM community.

MULTI-THREAD SOLUTIONS

The computationally hard problem of mining frequent patterns has been widely studied to propose viable solutions since it is certainly crucial for large-scale data (Aggarwal & Han, 2014). The simplest solutions related to parallel FIM were based on multi-thread architectures with shared memory. In other words, a single computer with multiple processors. Considering a data distribution among the processors or computer nodes (each node receives a data subset), the support counting could be easily computed for each data segment. However, the use of multi-thread solutions with shared memory has not been so important mainly because memory requirements are one of the main problems to be solved in FIM (Moens et al., 2013). It explains why solutions based on distributed computing, especially with the Big Data boom, have been much more numerous (Apiletti et al., 2017).

One of the first solutions based on a multi-thread architecture followed the Apriori methodology, dividing original data into segments and requiring to analyse all such segments before calculating a new level (candidate supersets) (Zaki, Parthasarathy, Ogihara, & Li, 1997). A different parallel version based on DIC, on the contrary, was proposed later (Cheung, Hu, & Xia, 1998). This parallel version was able to dynamically incorporate new itemsets since nodes can compute the support of potential frequent itemsets, once they get more results from other nodes, and make adjustments. In 2001, right after the FP-Growth algorithm was proposed (Han et al., 2000), the MLFPT (Multiple Local Frequent Pattern Tree) algorithm was presented (Zaïane, El-Hajj, & Lu, 2001), which is a parallel version of FP-Growth. Its main feature is the fair distribution of work among processors of a single computer.

Years later, with the increasing interest in graphics processing units (GPUs), some novel parallel GPU-based approaches were proposed. GPUs provide fast parallel hardware for a fraction of the cost of a traditional parallel system, consisting of a large number of processors and devices. Thus, in 2010, the GPU-FPM (Graphic Processing Unit to perform Frequent Pattern Mining) approach was presented to speed-up the arduous process of mining frequent itemesets (Zhou, Yu, & Wu, 2010). This approach is based on Apriori and, to solve GPU hardware delimitations, a compact data structure was designed to store the entire dataset on GPU. Right after that, a tree projection-based solution based on GPU was proposed (Teodoro, Mariano, Meira Jr., & Ferreira, 2010) since it has been already demonstrated that tree-projection techniques such as FP-Growth outperform Apriori-like approaches (Han et al., 2000). Recently, some authors (Djenouri, Djenouri, Belhadi, & Cano, 2018) have considered GPUs to obtain frequent itemsets in a single scan. Finally, in 2013, GPUs were used to speed up the evaluation phase (support counting in FIM) of non-exhaustive search FIM approaches. For any heuristic search approach this is generally one of the most time consuming phases (Ventura & Luna, 2016). Thanks to using a GPU, a high performance was achieved regardless of the number of solutions and data records to be evaluated (Cano, Luna, & Ventura, 2013). It is important to highlight that no new approaches have been proposed in this sense since the only achieved advance was related to the evaluation phase and, therefore, further improvements can be simply obtained with better hardware.

To sum up research studies about multi-thread solutions for FIM, Table 3 illustrates the most well-known algorithms among the research community. Proposed solutions generally aim at analysing a large number of data records by parallelizing the support counting process. Nevertheless, as it is demonstrated by the number of references and citations, multi-thread solutions cannot be considered as major innovation in the FIM field. In a general sense, these are shared-memory solutions that still suffer from memory requirements — one of the main problems in FIM — and they are typically not scalable enough. Thus, distributed computing solutions in which each computer node has its own memory are more numerous in this research field.

Algorithm	Type	Reference	Year	#Citations GoogleScholar
Parallel Apriori	Exhaustive	(Zaki et al., 1997)	1997	1,739
Parallel DIC	Exhaustive	(Cheung et al., 1998)	1998	52
MLFPT	Exhaustive	(Zaïane et al., 2001)	2001	190
GPU-FPM	Exhaustive	(Zhou et al., 2010)	2010	48
GPU-TreeProjection	Exhaustive	(Teodoro et al., 2010)	2010	23
GPU-Evaluation	Non-Exhaustive	(Cano et al., 2013)	2013	42
GPU-SingleScan	Exhaustive	(Djenouri et al., 2018)	2018	3

Table 3: List of the most important multi-thread FIM solutions (shared-memory) ordered by year. Last updated May 20th, 2019.

DISTRIBUTED COMPUTING SOLUTIONS

Parallel programming architectures can be grouped into shared memory and distributed computing approaches (where each processor share nothing, i.e. has private main memory and storage). In the previous section, existing FIM approaches on shared memory systems have been listed and described. This section, on the contrary, discusses FIM proposals for distributed systems (composed of processors that have their own internal memories and communicate with each other by passing messages). According to some authors, distributed systems allow quasi linear scalability and thus such approaches are becoming more and more common (Moens et al., 2013). An important factor explaining this popularity is the MapReduce framework proposed by Google (Dean & Ghemawat, 2008), which simplifies the programming for distributed data processing. Most of the existing distributed solutions for FIM are based on this framework even though some authors have asserted that the initial design principles of the MapReduce framework do not fit well for the FIM problem (Moens et al., 2013).

Most of the distributed computing solutions for FIM are based on the Partition algorithm (Savasere et al., 1995), a sequential solution proposed in 1995. This algorithm was proposed with the aim of improving the performance by reducing the number of disk I/Os. The algorithm splits data into chunks so that each chunk can be kept into main memory and frequent itemsets are obtained from each data subset. All these local frequent itemsets are combined to obtain the global frequency. Considering the features proposed by the sequential Partition algorithm, the FDM (Fast Distributed Mining of association rules) approach was proposed one year later (Cheung et al., 1996). Like Partition, FDM divides the whole dataset into data subsets and it then applies a classical Apriori algorithm to all the subsets (candidates for each level are computed in a distributed way). Once the candidate sets are obtained for a specific level, two pruning techniques are applied: local pruning and global pruning. FDM considers that every globally frequent itemset must be locally frequent in at least one node (a data subset). Additionally, if an itemset is locally frequent in a data subset, then all of its subsets are also locally frequent in that data subset. Also, if a globally frequent itemset is also frequent in a specific data subset, then all of its subsets behave in the same way (are globally frequent and locally frequent in that data subset).

It is not until 2008, once the MapReduce framework was released that the first FIM proposal based on MapReduce was presented, called PFP. From this date onwards, most of the existing proposals based on distributed computing considered the MapReduce framework. In 2012, three Apriori-like versions based on MapReduce were described SPC (Single Pass Counting), FPC (Fixed Passes Combined-Counting) and DPC (Dynamic Passes Counting) (Lin et al., 2012). SPC is a naive conversion of the Apriori algorithm for MapReduce. SPC can be considered as an adaptation of the FDM algorithm for the MapReduce framework. In each level of the Apriori algorithm, the support counts of candidate itemsets are computed through map (compute local frequencies for each itemset) and reduce (merge all the local frequencies to obtain the global frequency) procedures. Similar to Apriori, candidates that surpass the frequency threshold are passed as frequent itemsets for the map tasks of the next level. FPC, on the contrary, allows multiple consecutive passes or levels to be analysed on the same MapReduce iteration. Thus, for instance, in the same MapReduce iteration it is possible to obtain candidate itemsets of length 3, 4 and 5. The next iteration would then calculate itemsets of length 6, 7 and 8, and so on. As a result, FPC drastically reduces the number of passes over the dataset with regard to SPC (this latter requires one iteration per level). The main problem of FPC is that it can be overloaded by candidates. Thus, the DPC algorithm was proposed to balance the workloads between combined passes by dynamically merging candidates of several passes to maximize the utilization of each node during each MapReduce iteration. To do that, DPC uses a threshold to indicate the maximum number of candidates for counting in a mapper so DPC continuously generates and collects candidates of longer length until the total number of candidates is larger than that threshold.

It was in 2013 that an important contribution was provided through two new methods for FIM on the MapReduce framework (Moens et al., 2013). Unlike most of existing MapReduce solutions, these methods were based on the efficient ECLAT algorithm. The first method, named Dist-Eclat (Moens et al., 2013), divides the search space rather than the data space and, hence reducing the required communication to obtain global frequencies compared to most approaches so far. Therefore, no extra communication between mappers is necessary and no checking of overlapping mining results has to be done. This algorithm based on

ECLAT, and more specifically on ECLAT using diffsets, first computes the singletons by a MapReduce process. Then, in a second step and considering the frequent singletons, frequent itemsets of length k are generated by running ECLAT up to level k. The final step consists of mining the prefix tree using a prefix assigned in the previous step. The second method, called BigFIM (Moens et al., 2013), is a hybrid method that first uses an Apriori based approach to extract frequent itemsets of length k and later on switches to ECLAT when the projected dataset fits in memory.

Whilst the number of MapReduce approaches for FIM has been numerous (L. Wang, Feng, Zhang, & Liao, 2014; C. Wang, Lin, & Chang, 2017), existing methods still do not have a good scalability due to high workload skewness, large intermediate data, and large network communication overhead. In this sense, the BIGMiner algorithm (Chon & Kim, 2018) was recently proposed, which was described as a fast and scalable MapReduce proposal for FIM. According to the authors, BIGMiner is a highly scalable proposal because it has no workload skewness, no intermediate data, and small network communication overhead. Other recent MapReduce solutions (Luna, Padillo, et al., 2018), based on the anti-monotone property, have been also proposed in this regard.

Although the FIM problem is generally solved through exhaustive search methodologies, it has been recently solved through non-exhaustive (heuristic) search methodologies based on MapReduce. In such proposals, it is the evaluation process that requires to be parallelized since it is the most time consuming part. It is important to recall that heuristic solutions gained interest among researchers due to their ability to cope with continuous features without a discretization step even though they may miss some part of the search space. The first non-exhaustive approach for MapReduce is the MRQAR algorithm (Martín et al., 2018), which follows a generic parallel framework to discover quantitative association rules on Big Data. Another interesting non-exhaustive approach based on MapReduce was recently proposed (Padillo, Luna, Herrera, & Ventura, 2018). This approach, known as G3P-LSC, is based on a grammar-guided genetic programming methodology and uses the MapReduce framework to evaluate the generated solutions.

As a summary of distributed computing solutions for FIM, Table 4 shows the most well-known algorithms among the research community. PFP, Dist-Eclat and BigFIM appear in

Algorithm	Type	Reference	Year	#Citations GoogleScholar
FDM	Exhaustive	(Cheung et al., 1996)	1996	660
PFP	Exhaustive	(H. Li et al., 2008)	2008	456
SPC	Exhaustive	(Lin et al., 2012)	2012	202
FPC	Exhaustive	(Lin et al., 2012)	2012	202
DPC	Exhaustive	(Lin et al., 2012)	2012	202
Dist-Eclat	Exhaustive	(Moens et al., 2013)	2013	164
BigFIM	Exhaustive	(Moens et al., 2013)	2013	164
SPAprioriMR	Exhaustive	(Luna, Padillo, et al., 2018)	2018	7
BIGMiner	Exhaustive	(Chon & Kim, 2018)	2018	3
MRQAR	Non-Exhaustive	(Martín et al., 2018)	2018	4
G3P-LSC	Non-Exhaustive	(Padillo et al., 2018)	2018	13

Table 4: List of the most important distributed computing FIM solutions (share nothing) ordered by year. Algorithms highlighted in gray denote key solutions in the field (those used in most of recent comparative analyses). Last updated May 20th, 2019.

most of the comparative studies in FIM and they are, by far, the key approaches in the field. Few approaches are related to non-exhaustive search methodologies and it is mainly because distributed computing solutions almost solve the runtime and memory requirement problems (when a huge number of computer nodes is considered) and, therefore, exhaustive search models that analyse the whole search space are much more appropriate. As a result, and according to the number of citations, it is demonstrated that exhaustive search methods are desirable for the FIM community when the MapReduce framework is used.

LESSON LEARNED

In this article we have listed the problems that arise in the FIM task and how different authors have proposed wide and diverse solutions to address these problems. The aim of this article is to show the improvements addressed during the last 25 years, that is, since the FIM task was first described (Agrawal et al., 1993). While some reviews have been already proposed in literature (Fournier-Viger et al., 2017; Chee, Jaafar, Aziz, Hasan, &

Yeoh, 2018) they are mainly focused on sequential exhaustive search approaches and on describing the algorithms for non-expert users. In this sense, it is our understanding that an analysis from an expert point of view that involves any existing methodology (exhaustive and non-exhaustive search) on any architecture (centralized and parallel) is necessary to comprehend which improvements have been proposed over time. It is noteworthy that the mere fact of understanding how or why a recent algorithm takes some algorithmic concepts of previous approaches might be the key fact for new ideas flowing.

Around the ealier years in which the FIM problem was first described (Agrawal et al., 1993), a series of solutions were proposed (Aggarwal & Han, 2014) which were generally based on the Apriori algorithm, the first approach for mining frequent patterns and, nowadays, the most-well known algorithm. In other words, any solution proposed at the beginning followed a levelwise breadth first search methodology that produced candidate itemsets whose frequency counting was performed by reading the dataset multiple times (one for each size of candidate itemsets). It is important to remark that, although really small improvements were achieved during these first years, features of such algorithms have been the basis for current and trending distributed computing solutions. A clear example of that is the Partition algorithm (Savasere et al., 1995), which is being cited nowadays even more than in the years after its publication. The idea of Partition was to split data into chunks so that they are kept into main memory and local frequent itemsets were obtained from each data subset and, it is the methodology used by modern MapReduce based approaches (Moens et al., 2013).

Analysing the literature and considering not only comparative analyses carried out but also the number of citations, only three important improvements with regard to the baseline (Apriori) for centralized algorithms have been proposed over time. The first one, called FP-Growth (Han et al., 2000), did not follow a levelwise methodology as Apriori and achieved important reductions in runtime. The second one, called ECLAT (Zaki, 2000), was able to be run in a runtime smaller than FP-Growth but just in some cases —if no extremely large lists of transactions were required to be analysed. A great improvement in runtime (several orders of magnitude) with regard to ECLAT on dense datasets was achieved by dECLAT (Zaki & Gouda, 2003). The third one is LCM, which won the FIMI'04 competition and it is, nowadays, one of the fastest algorithm for most of the problems. After the two competitions

(FIMI'03 and FIMI'04), algorithms were so good that it is hard to find any solutions with a higher performance. Really efficient implementations of these algorithms can be found in the literature (Borgelt, 2003). At this point it should be highlighted that, in general terms, the FIM community is more focused on exhaustive search solutions than on non-exhaustive search solutions (they are not so widely used as the number of citations indicates in Table 2). It is our belief that the reason is based on the inability of non-exhaustive methods to guarantee the whole search space to be explored. Nevertheless, heuristic methodologies in FIM provide really interesting features: ability to cope with numerical attributes and a runtime that is almost constant regardless of the number of items in data.

Considering some of the ideas proposed during the years right after the FIM problem definition (Agrawal et al., 1993), some parallel (shared-memory) approaches were presented (Zaki et al., 1997; Zaïane et al., 2001). However, these multi-thread solutions (see Table 3) with shared memory were not so important since one of the main problems of FIM is to cope with large amount of data in memory (a high number of items in data produce an extremely large set of itemsets) and, therefore, the use of multiple processors sharing the memory is not an optimal solution. Not much attention was paid on these methods till 2010, when computing based on GPUs started to be an option for hard computational problems. In this sense, different solutions based on GPU were proposed not only for exhaustive search (Zhou et al., 2010) but also for non-exhaustive search (Cano et al., 2013) methodologies. It was in 2008 when first distributed computing solutions —each compute node being completely independent and each having its own memory— were proposed (see Table 4). These approaches, based on MapReduce, considered chunks of the original dataset to be analysed in each compute node in a similar fashion as it was proposed in the Partition algorithm in 1995 (Savasere et al., 1995). MapReduce solutions based on Apriori (Lin et al., 2012), FP-Growth (H. Li et al., 2008) and Eclat (Moens et al., 2013) can be found in the literature. Similarly, some non-exhaustive search methods have also been proposed (Martín et al., 2018). It is our understanding that even when extremely efficient solutions based on MapReduce have been proposed, none of them propose a completely new methodology (solutions are mainly based on applications of existing algorithms to the MapReduce framework). It is also remarkable that such improvements were obtained even when some authors have asserted that the initial design principles of the MapReduce framework do not fit well for the FIM problem (Moens et al., 2013).

On the basis of all the above, it is interesting to highlight that few real (innovative) solutions can be found in the FIM literature. In general terms, until now, the four main exhaustive-search algorithms described in the literature are Apriori (Agrawal et al., 1993), FP-Growth (Han et al., 2000), ECLAT (Zaki, 2000) and dECLAT (Zaki & Gouda, 2003). All of them have their own features and for this reason, are the most-well known solutions in the field. All the other existing solutions are mainly adaptations of these four algorithms, even those currently proposed and based on the MapReduce framework. As for non-exhaustive search solutions, multiple algorithms can be found in the literature. However, they are not so important in the FIM research community since they do not guarantee that the whole search space is explored.

Analyzing the specialized literature and considering any existing algorithm for mining frequent itemsets, there is not necessarily a clear winner in terms of performance for all cases and some studies may even have contradictory conclusions. Some problems for doing a fair evaluation is that many implementations are not public, and authors do not use the same datasets, the same programming language, or they implement the competitor algorithms (the skills of the programmer play an important rol), which is a controversial issue as it has been recently described (Kriegel, Schubert, & Zimek, 2017). Nevertheless, FIMI competitions (FIMI'03 and FIMI'04) have provided a good snapshot of the performance of algorithms at that time. In general terms, LCM (Uno et al., 2004) and SaM (Borgelt, 2010) best perform for most of the cases and it is hard to decide which should be used in any situation.

Finally, considering future research directions of FIM, a recent study (Luna, 2016) based on emerging topics in the field described that comprehensibility (any itemset can be little comprehensible for a specific user and, at the same time, very comprehensible for a different one) and flexibility (ability of adapting the solutions to the user's requirements by introducing subjective knowledge into the mining process) might be two challenging research issues.

FUNDING INFORMATION

This research was supported by the Spanish Ministry of Economy and Competitiveness and the European Regional Development Fund, projects TIN2017-83445-P.

References

Aggarwal, C. C., & Han, J. (2014). *Frequent pattern mining.* Springer Publishing Company, Incorporated.

Aggarwal, C. C., & Yu, P. S. (2001). On effective conceptual indexing and similarity search in text data. In *Proceedings of the 2001 IEEE International Conference on Data Mining (ICDM '01)* (pp. 3–10). San Jose, California, USA.

Agrawal, R., Imielinski, T., & Swami, A. N. (1993). Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data (ICDM '93)* (pp. 207–216). Washington, DC, USA.

Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB '94)* (pp. 487–499). San Francisco, CA, USA.

Alatas, B., & Akin, E. (2006). An efficient genetic algorithm for automated mining of both positive and negative quantitative association rules. *Soft Computing*, *10*, 230–237.

Apiletti, D., Baralis, E., Cerquitelli, T., Garza, P., Pulvirenti, F., & Venturini, L. (2017). Frequent Itemsets Mining for Big Data: A Comparative Analysis. *Big Data Research*, *9*, 67–83.

Bansal, D., & Bhambhu, L. (2014). Usage of apriori algorithm of data mining as an application to grievous crimes against women. *International Journal of Computer Trends and Technology*, *4*(9), 3194–3199.

Borgelt, C. (2003). Efficient implementations of apriori and eclat. In *Proceedings of the 1st IEEE ICDM Workshop on Frequent Item Set Mining Implementations (FIMI '03).* Melbourne, Florida, USA.

Borgelt, C. (2005). Keeping things simple: Finding frequent item sets by recursive elimination. In *Proceedings of the 1st International Workshop on Open Source Data Mining: Frequent Pattern Mining Implementations (OSDM '05)* (pp. 66–70). Chicago, Illinois.

Borgelt, C. (2010). Simple algorithms for frequent item set mining. In *Advances in Machine Learning II, Dedicated to the Memory of Professor Ryszard S. Michalski* (pp. 351–369).

Borgelt, C. (2012). Frequent item set mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *2*(6), 437–456.

Brin, S., Motwani, R., Ullman, J. D., & Tsur, S. (1997). Dynamic itemset counting and implication rules for market basket data. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data* (pp. 255–264). Tucson, Arizona, USA.

Cai, C. H., Fu, A. W., Cheng, C. H., & Kwong, W. W. (1998). Mining association rules with weighted items. In *Proceedings of the 1998 International Database Engineering and Applications Symposium (IDEAS '98)* (pp. 68–77). Cardiff, Wales, UK.

Cano, A., Luna, J. M., & Ventura, S. (2013). High performance evaluation of evolutionary-mined association rules on gpus. *The Journal of Supercomputing*, *66*(3), 1438-1461.

Chee, C.-H., Jaafar, J., Aziz, I. A., Hasan, M. H., & Yeoh, W. (2018). Algorithms for frequent itemset mining: a literature review. *Artificial Intelligence Review. In press.* doi: 10.1007/s10462-018-9629-z

Cheung, D. W., Han, J., Ng, V. T., Fu, A. W., & Fu, Y. (1996). A fast distributed algorithm for mining association rules. In *Proceedings of the 4th International Conference on on Parallel and Distributed Information Systems (DIS '96)* (pp. 31–43). Miami Beach, Florida, USA.

Cheung, D. W., Hu, K., & Xia, S. (1998). Asynchronous parallel algorithm for mining association rules on a shared-memory multi-processors. In *Proceedings of the 10th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA '98)* (pp. 279–288). Puerto Vallarta, Mexico.

Chon, K., & Kim, M. (2018). Bigminer: a fast and scalable distributed frequent pattern miner for big data. *Cluster Computing*, *21*(3), 1507–1520.

Dean, J., & Ghemawat, S. (2008). Mapreduce: simplified data processing on large clusters.

Communications of the ACM, *51*(1), 107–113.

Dehaspe, L., Toivonen, H., & King, R. D. (1998). Finding frequent substructures in chemical compounds. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98)* (pp. 30–36). New York City, New York, USA.

Djenouri, Y., Djenouri, D., Belhadi, A., & Cano, A. (2018). Exploiting gpu and cluster parallelism in single scan frequent itemset mining. *Information Sciences, In press*. doi: 10.1016/j.ins.2018.07.020

Fournier-Viger, P., Lin, J. C., Vo, B., Truong, T. C., Zhang, J., & Le, H. B. (2017). A survey of itemset mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *7*(4).

Fournier-Viger, P., Chun-Wei Lin, R., Jerry Nkambou, Vo, B., & Tseng, V. S. (2019). *High-Utility Pattern Mining*. Springer International Publishing.

Fournier-Viger, P., Lin, J. C.-W., Kiran, R. U., & Koh, Y. S. (2017). A survey of sequential pattern mining. *Data Science and Pattern Recognition*, *1*(1), 54–77.

Gama, J. (2010). *Knowledge discovery from data streams*. CRC Press.

Gamberger, D., & Lavrac, N. (2002). Expert-guided subgroup discovery: Methodology and application. *Journal of Artificial Intelligent Resesearch*, *17*(1), 501–527.

Gan, W., Lin, J. C., Fournier-Viger, P., Chao, H., Hong, T., & Fujita, H. (2018). A survey of incremental high-utility itemset mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *8*(2).

Guil, F., Juárez, J. M., & Marín, R. (2007). A possibilistic approach for mining uncertain temporal relations from diagnostic evolution databases. In *Proceedings of the 2nd International Work-Conference on the Interplay Between Natural and Artificial Computation, (IWINAC '07)* (pp. 597–606). La Manga del Mar Menor, Spain.

Han, J., & Kamber, M. (2000). *Data mining: Concepts and techniques*. Morgan Kaufmann.

Han, J., Pei, J., & Yin, Y. (2000). Mining frequent patterns without candidate generation. *SIGMOD Rec.*, *29*(2), 1–12.

Jiang, C., Coenen, F., & Zito, M. (2013). A survey of frequent subgraph mining algorithms. *The Knowledge Engineering Review*, *28*(1), 75–105.

Koh, Y. S., & Ravana, S. D. (2016). Unsupervised rare pattern mining: a survey. *ACM*

Transactions on Knowledge Discovery from Data, *10*(4), 45.

Kraska, T. (2013). Finding the needle in the big data systems haystack. *IEEE Internet Computing*, *17*(1), 84–86.

Kriegel, H., Schubert, E., & Zimek, A. (2017). The (black) art of runtime evaluation: Are we comparing algorithms or implementations? *Knowledge and Information Systems*, *52*(2), 341–378.

Li, H., Wang, Y., Zhang, D., Zhang, M., & Chang, E. Y. (2008). Pfp: Parallel fp-growth for query recommendation. In *Proceedings of the 2008 ACM Conference on Recommender Systems (RecSys '08)* (pp. 107–114). Lausanne, Switzerland.

Li, J., & Wong, L. (2002). Identifying good diagnostic gene groups from gene expression profiles using the concept of emerging patterns. *Bioinformatics*, *18*(10), 1406–1407.

Lin, M.-Y., Lee, P.-Y., & Hsueh, S.-C. (2012). Apriori-based frequent itemset mining algorithms on mapreduce. In *Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication (ICUIMC '12)* (pp. 1–8). Kuala Lumpur, Malaysia.

Luna, J. M. (2016). Pattern mining: current status and emerging topics. *Progress in AI*, *5*(3), 165–170.

Luna, J. M., Cano, A., Pechenizkiy, M., & Ventura, S. (2016). Speeding-up association rule mining with inverted index compression. *IEEE Transactions on Cybernetics*, *46*(12), 3059–3072.

Luna, J. M., Ondra, M., Fardoun, H. M., & Ventura, S. (2018). Optimization of quality measures in association rule mining: an empirical study. *International Journal of Computational Intelligence Systems*, *12*, 59–78.

Luna, J. M., Padillo, F., Pechenizkiy, M., & Ventura, S. (2018). Apriori versions based on mapreduce for mining frequent patterns on big data. *IEEE Transactions on Cybernetics*, *48*(10), 2851–2865.

Luna, J. M., Romero, J. R., Romero, C., & Ventura, S. (2014). Reducing gaps in quantitative association rules: A genetic programming free-parameter algorithm. *Integrated Computer-Aided Engineering*, *21*(4), 321–337.

Luna, J. M., Romero, J. R., & Ventura, S. (2012). Design and behavior study of a grammar-guided genetic programming algorithm for mining association rules. *Knowledge and Information Systems*, *32*(1), 53–76.

Martín, D., Martínez-Ballesteros, M., García-Gil, D., Alcalá-Fdez, J., Herrera, F., & Santos, J. C. R. (2018). MRQAR: A generic mapreduce framework to discover quantitative association rules in big data problems. *Knowledge-Based Systems*, *153*, 176–192.

Mata, J., Alvarez, J. L., & Riquelme, J. C. (2001). Mining numeric association rules with genetic algorithms. In *Proceedings of the 5th International Conference on Artificial Neural Networks and Genetic Algorithms* (pp. 264–267). Taipei, Taiwan.

Moens, S., Aksehirli, E., & Goethals, B. (2013). Frequent itemset mining for big data. In *Proceedings of the 2013 IEEE International Conference on Big Data* (pp. 111–118). Santa Clara, CA, USA.

Noaman, A. Y., Luna, J. M., Ragab, A. H. M., & Ventura, S. (2016). Recommending degree studies according to students' attitudes in high school by means of subgroup discovery. *International Journal of Computational Intelligent Systems*, *9*(6), 1101–1117.

Padillo, F., Luna, J. M., Herrera, F., & Ventura, S. (2018). Mining association rules on big data through mapreduce genetic programming. *Integrated Computer-Aided Engineering*, *25*(1), 31–48.

Park, J. S., Chen, M.-S., & Yu, P. S. (1995). An effective hash-based algorithm for mining association rules. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data* (pp. 175–186). San Jose, California, USA.

Pei, J., Han, J., Lu, H., Nishio, S., Tang, S., & Yang, D. (2001). H-mine: Hyper-structure mining of frequent patterns in large databases. In *Proceedings of the 2001 IEEE International Conference on Data Mining (ICDM '01)* (pp. 441–448). San Jose, California, USA.

Romero, C., Zafra, A., Luna, J. M., & Ventura, S. (2013). Association rule mining using genetic programming to provide feedback to instructors from multiple-choice quiz data. *Expert Systems*, *30*(2), 162–172.

Ruggieri, S., Pedreschi, D., & Turini, F. (2010). Data mining for discrimination discovery. *ACM Transactions on Knowledge Discovery from Data*, *4*(2), 9:1–9:40.

Salleb-Aouissi, A., Vrain, C., & Nortet, C. (2007). Quantminer: A genetic algorithm for mining quantitative association rules. In *Proceedings of the 20th International Conference on Artificial Intelligence* (p. 1035-1040). Hyberadad, India.

Savasere, A., Omiecinski, E., & Navathe, S. (1998). Mining for strong negative associations in a large database of customer transactions. In *Proceedings of the 14th International Conference on Data Engineering* (pp. 494–502). Orlando, Florida, USA.

Savasere, A., Omiecinski, E., & Navathe, S. B. (1995). An efficient algorithm for mining association rules in large databases. In *Proceedings of the 21th International Conference on Very Large Data Bases (VLDB '95)* (pp. 432–444). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Srikant, R., & Agrawal, R. (1995). Mining generalized association rules. In *Proceedings of the 21st International Conference on Very Large Data Bases (VLDB '95)* (pp. 407–419). Zurich, Switzerland.

Tan, P., Kumar, V., & Srivastava, J. (2004). Selecting the right objective measure for association analysis. *Information Systems*, *29*(4), 293–313.

Teodoro, G., Mariano, N., Meira Jr., W., & Ferreira, R. (2010). Tree projection-based frequent itemset mining on multicore cpus and gpus. In *Proceedings of the 22nd International Symposium on Computer Architecture and High Performance Computing* (pp. 47–54).

Uno, T., Kiyomi, M., & Arimura, H. (2004). LCM ver. 2: Efficient Mining Algorithms for Frequent/Closed/Maximal Itemsets. In *Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI '04).* Brighton, UK.

Ventura, S., & Luna, J. M. (2016). *Pattern Mining with Evolutionary Algorithms.* Springer International Publishing.

Ventura, S., & Luna, J. M. (2018). *Supervised descriptive pattern mining.* Springer Publishing Company, Incorporated.

Wang, C., Lin, S., & Chang, J. (2017). Mapreduce-based frequent pattern mining framework with multiple item support. In *Proceedings of the 2017 Asian Conference on Intelligent Information and Database Systems* (pp. 65–74). Kanazawa, Japan.

Wang, H., Zhang, X., & Chen, G. (2008). Mining a complete set of both positive and negative association rules from large databases. In *Proceedings of the 12th Pacific-Asia*

Conference on Knowledge Discovery and Data Mining (PAKDD '08) (pp. 777–784). Osaka, Japan.

Wang, L., Feng, L., Zhang, J., & Liao, P. (2014). An efficient algorithm of frequent itemsets mining based on mapreduce. *Journal of Information & Computational Science*, *11*(8), 2809–2816.

Yan, X., Zhang, C., & Zhang, S. (2009). Genetic algorithm-based strategy for identifying association rules without specifying actual minimum support. *Expert Systems with Applications*, *36*(2), 3066–3076.

Zaïane, O. R., El-Hajj, M., & Lu, P. (2001). Fast parallel association rule mining without candidacy generation. In *Proceedings of the 2001 IEEE International Conference on Data Mining (ICDM '01)* (pp. 665–668). San Jose, California, USA.

Zaki, M. J. (2000). Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, *12*(3), 372–390.

Zaki, M. J., & Gouda, K. (2003). Fast vertical mining using diffsets. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 326–335). Washington, D.C. USA.

Zaki, M. J., Parthasarathy, S., Ogihara, M., & Li, W. (1997). Parallel algorithms for discovery of association rules. *Data Mining and Knowledge Discovery*, *1*(4), 343–373.

Zhang, C., & Zhang, S. (2002). *Association rule mining: models and algorithms.* Springer Berlin / Heidelberg.

Zheng, Y. (2015). Trajectory data mining: an overview. *ACM Transactions on Intelligent Systems and Technology*, *6*(3), 29.

Zhou, J., Yu, K., & Wu, B. (2010). Parallel frequent patterns mining algorithm on GPU. In *Proceedings of the 2010 IEEE International Conference on Systems, Man and Cybernetics* (pp. 435–440). Istanbul, Turkey.