Categorization of Association Rule Mining Algorithms

Yanbo Wang

Department of Computer Science, University of Liverpool
Liverpool L69 7ZF, United Kingdom
{jwang}@csc.liv.ac.uk

Abstract

More and more computer science scholars and researchers, especially those who specialize in the field of Knowledge Discovery in Data (KDD), focus and emphasis on Association Rule Mining (ARM). ARM, arguably, is one of the most researched areas in KDD, addresses the problem of discovering association rules between items / attributes in very large databases. A number of significant ARM algorithms have been proposed. At the same time, different applications for ARM have been identified. Further, a number of ARM extensions and variations have appeared to address different real-life problems. In this paper, a summary / survey of the studies in ARM, that includes descriptions and a classification of the most common ARM algorithm, is presented with the aim of supporting future work in KDD research.

1. INTRODUCTION

Nowadays, Association Rule Mining (ARM) is one of the major techniques / tasks in data mining (the main stage of Knowledge Discovery in Data). ARM has already been widely applied in many different areas since it was first introduced in [1]. One typical application [2], "shopping basket analysis" or "market basket analysis", focuses on the discovery of interesting hidden patterns (associations) between items or combinations of items in supermarket transaction records. Some other applications in ARM have also received a great deal of attention from many KDD researches, such as (i) student-institution academic enrolment, (ii) client-server web pages exploration, and (iii) the analysis text-document word occurrences [11]. Further applications include the identification of interesting relations within stock investment and portfolio management data, and mining collections of textual documents for interesting knowledge [19], etc.

The structure of this paper is organized as follows. In Section 2, we review the principle of ARM. In Section 3, we summarize the three major conceptual measure frameworks in ARM. In Section 4, we categorize the existing ARM algorithms based on their different nature using the idea of "families" of ARM algorithms. We finally present some conclusions in Section 5.

2. SOME ARM DEFINITIONS

From the above the central theme of ARM is the identification of interesting relationships in data. The input data *(D)* for most ARM algorithms comprises N columns describing a binary valued set of attributes A, and M transactions such that each transaction describes some subset of A. The relationships of interest are expressed in rule form:

$$X \rightarrow Y$$

Where X (the **antecedent**) and Y (the **consequent**) are disjoint subsets of A. The rule $X \rightarrow Y$ is interpreted as "if X exists it is likely that Y also exists".

Any subset I of the available set of attributes *(A)* is called **an item set**. In the above discussion X, Y and $X \cup Y$ are all item sets. The length of an item set *(|I|)* is equivalent to the number of items in an item set. An itemset of length k is often referred to as **a *k*-itemset**. The last n attributes of any

item set *I* is referred to as the **n-extension of *I***.

3. CONCEPTUASL MEASURES

"Interestingness" is the most significant measures of prediction-rule quality in ARM. The most well known framework is the Support/Confidence Framework. Many sequential and parallel ARM algorithms refer to this framework.

Support/confidence focuses on two percentage values, where support *(σ)* represents the frequency of an item set, and confidence *(α)* represents the "strength" of a rule. The support for an item set *X* is defined determined using the identity [22]:

$$\sigma(X) = \frac{\text{numberOf Transactions In whichX occurs}}{M}$$

(Where *M* is the total number of transactions in the Dataset, *D*. The confidence in a rule $X \rightarrow Y$ is then calculated as follows:

$$\alpha(X \rightarrow Y) = \frac{\sigma\ (X \cup Y)}{\sigma\ (X)}$$

If the support of an item set is greater than a given support threshold, *"min_sup"*, the item set is said to be large or frequent. Rules are generated from frequent item sets. If the confidence of a rule exceeds a chosen confidence threshold, *"min_conf"*, the rule said to be "strong".

A further indicator of the relevance of a rule, that is sometime applied, is lift *(δ)*. Lift [9] is an indicator of the "surprisingness" of a rule. An AR is considered surprising (and therefore interesting) only if its lift exceeds *"1"*. The lift for a rule $X \rightarrow Y$ is calculated as follows:

$$\delta(\mathrm{X} \rightarrow \mathrm{Y}) = \frac{\alpha(X \rightarrow Y) / \alpha(X)}{\alpha(Y)} = \frac{\sigma\ (X \rightarrow Y)}{\alpha(\mathrm{Y})}$$

A criticism of the Support/Confidence framework is presented in [19] where it is argued that confidence is unable to detect either statistical independence or negative dependence between antecedent and consequent. In other words when $\sigma(Y)$ is very high and $\alpha(X=>Y) > \sigma(Y)$, it is likely that a significant number of misleading rules, with very high confidence, will be reported. In recognition of this criticism a number of alternative "interestingness" frameworks have been proposed; two of these, (i) the Certainty Factors Framework and (ii) the Very Strong Rules Framework, are briefly reviewed in the following two sub-sections.

3.1 Certainty Factors Framework

The Certainty Factors Framework makes use of a certainty factor *(θ)*. The main concept of this framework is described as follows:

IF $\alpha(X=>Y) > \sigma(Y)$	*THEN* $\theta(X=>Y) = (\alpha(X=>Y) - \sigma(Y)) / (1-\sigma(Y))$
ELSE IF $\alpha(X=>Y) < \sigma(Y)$	*THEN* $\theta(X=>Y) = (\alpha(X=>Y) - \sigma(Y)) / \sigma(Y)$
ELSE	$\theta(X=>Y) = 0$

The range of θ takes values in between "-1" and "1", where "-1" (the maximum decrement) means that when *X* is true then *Y* is false, and "1" (the maximum increment) means that when *X* is true then *Y* is true. This framework is able to check the statistical independence (i.e., when $\theta(X=>Y) =$

0 or $\alpha(X=>Y) = \sigma(Y)$). It is also possible to detect the negative dependence (i.e., when $\theta(X=>Y) < 0$).

3. 2 Very Strong Rules Framework

The Very Strong Rules Framework introduced in [19], and discussed in [10], is based on the logically equivalence of "$X=>Y \equiv \neg Y=>\neg X$". Therefore, the logical formula of the Very Strong Rule *(ν)* is:

> $\nu(X=>Y)$ *iff* $(X=>Y) \equiv$ *Strong Rule* && $(\neg Y=>\neg X) \equiv$ *Strong Rule*

In fact, the point of this framework is "logic equivalence / conversion". It was first proposed in [11] that "$A=>B$" could be rewritten as "$\neg(A \wedge \neg B)$", named conviction *(β)*. Thus, $\beta = \sigma(X)\sigma(\neg Y) / \sigma(X=>\neg Y)$. The idea of β initially related and represented the consideration of ν. To Explain the logic of Very Strong Rules in depth: when $\sigma(Y)$ or $\sigma(X)$ is very high, $\sigma(\neg Y=>\neg X)$ is low, and hence the rule $(\neg Y=>\neg X)$ will not be strong and $(X=>Y)$ will not be strong. There are two short cuts:

> (1) *LET* $\sigma(X)+\sigma(Y)>=1$. *THEN* $X=>Y$ *is* ν *iff* $X=>Y$ *is strong.*
> (2) *LET* $\sigma(X)+\sigma(Y) < 1$. *THEN* $X=>Y$ *is* ν *iff* $\neg Y=>\neg X$ *is strong.*

In order to easily employ the three above frameworks in measuring association rules, two further hybrid properties have been summarized:

> (1) *IF* $\theta(X=>Y) > 0$ *THEN* $\theta(X=>Y) = \theta(\neg Y=>\neg X) > 0$
> (2) $\sigma(\neg Y=>\neg X) = 1 - \sigma(Y) - \sigma(X) + \sigma(X=>Y)$

4. CATEGORISATION OF ARM ALGORITHMS INTO "FAMILIES"

The central theme of this paper is the categorisation of existing ARM algorithms into families. Six families are proposed as listed below. Broadly these families have a chronological ordering associated with them and thus can also be interpreted as representing generations of ARM algorithms.

1. **Semi-Apriori-Like (1993-1995):** ARM algorithms that generated all frequent item sets but do not apply the Apriori generate-prune iterative approach founded on the downward closure property of item sets --- *if an item set is a frequent itemset then all its subsets will also be frequent item sets.*
2. **Pure-Apriori-Like (1994-1996):** ARM algorithms that generated all frequent itemsets and make use if the generate-prune level by level iteration approach first promulgated in the Apriori algorithm [2].
3. **Dynamic-Like (1995-2001):** ARM algorithms that generates all frequent keyword-sets with addressing a dynamical ideology and methodology.
4. **Maximal-Like (1993-1995):** ARM algorithms that generates all frequent itemsets but only partially applies the principle of Apriori.
5. **Tree-Structure-Like (1997-Present):** ARM algorithms that, as a first step, are directed the discovery of Maximal Frequent Item sets (MFIs) (a definition of MFI is presented in sub-section 4.5 below).
6. **Closed-Like (1999-Present)**: ARM algorithms that, in the first instance, are directed at generating the frequent closed item sets only.

The relationship between the different families is depicted in Figure 1. More detail is provided in the following sub-sections.

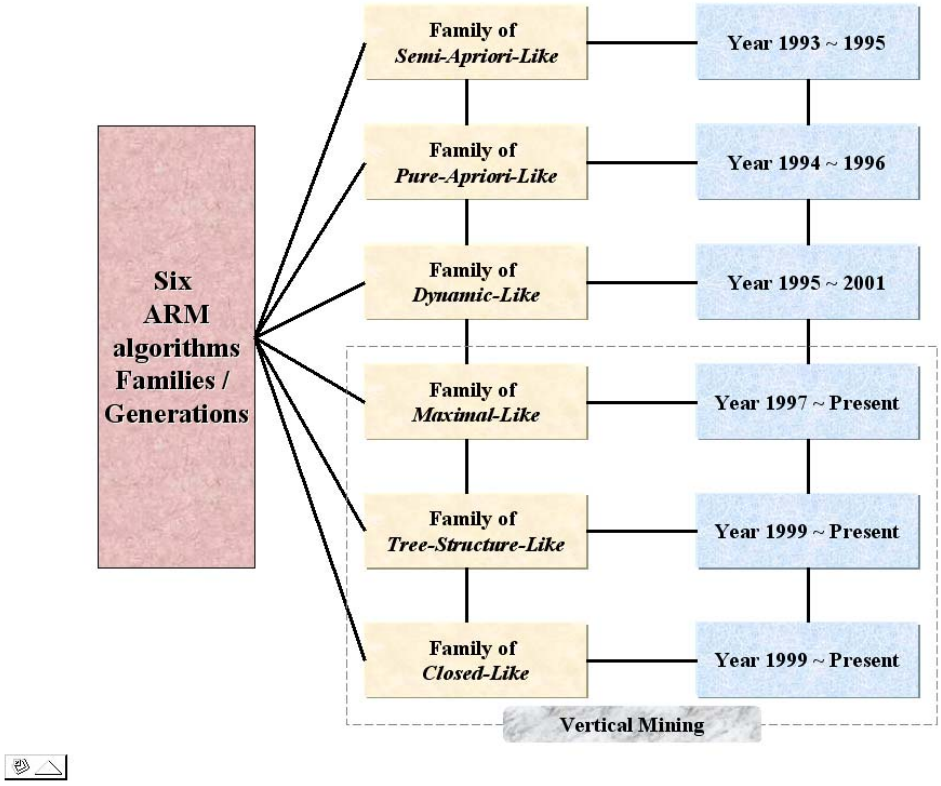

Figure 1: Categorization of ARM algorithms

4. 1 Family of Semi-Apriori-Like

The idea of Apriori is the core of many existing ARM algorithms. Apriori makes use of the principle that *"every single subset of a frequent itemset must also be frequent."* [2]. Apriori operates in an iterative manner as follows:

```
K=1
Generate all K-item sets
While (candidate K-item sets exist) {
     1. Determine support for candidate K-item sets from data.
     2. Remove all candidate K item sets that are not
        sufficiently supported to give frequent K-item sets.
     3. Generate K+1 itemsets from frequent K-item sets using
        downward closure property.
     4. K=K+1
     }
```

The family of *Semi-Apriori-Like* ARM algorithms precedes the Apriori algorithm although they do make use of some of the ideas that were eventually incorporated into Apriori. *Semi-Apriori-Like* algorithms tend not to immediately prune the non-frequent candidate item sets. The family includes the following exemplar algorithms:

1. **AIS** [1]
2. **SETM** (SET Oriented Mining) [32] [33] [34]
3. **OCD** (Offine Candidate Determination) [41]

(Figure 2).

SETM was latter used as the foundation for work on extensions of the DB query language SQL

such as the MINE RULE operator used in association with SQL SELECT statements [42]. ARM using SQL extensions has also been used in connection with (i) evolutionary computation techniques [55] and (ii) both horizontal and vertical versions of K-Way-Join and the algorithms of SCD and SCJ [52].

OCD operates in a very similar manner to the Apriori algorithm except in the generation of candidate item sets.

The family of *Semi-Apriori-Like* ARM algorithms display two significant disadvantages:

1. Generation of candidate itemsets
2. Multiple passes/scans of the database --- the number of passes is equivalent to the length of the largest frequentset plus one.

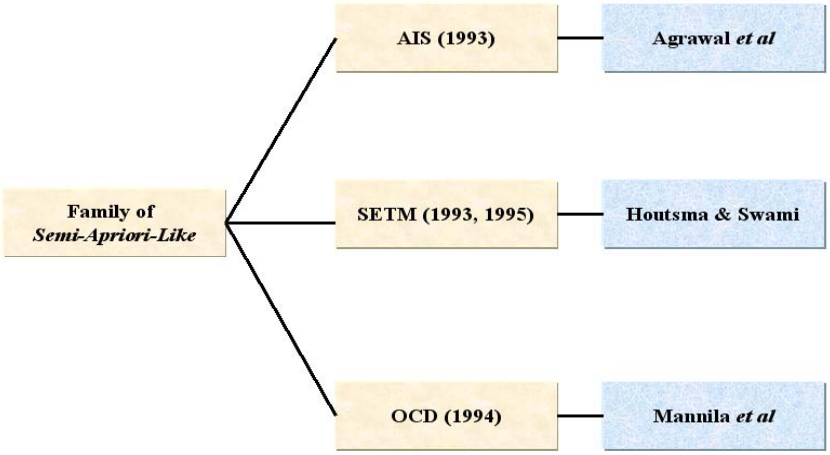

Figure 2: Family of Semi-Apriori-Like

4. 2 Family of Pure-Apriori-Like

The second category of ARM algorithms, *Pure-Apriori-Like*, began (as the name suggests) with the introduction (1994) of three similar algorithms [2], Apriori, AprioriTid and AprioriHybrid. This family can be considered to comprise two sub-branches:

1. Sequential algorithms
2. Parallel/distributed algorithms

(Figure 3)

The sequential branch can be further subdivided into: (a) complete, (b) partitioned, and (c) sample algorithms. The parallel/distributed branch, in turn, can be further subdivided into three different approaches: (a) data [45] [3] [13] [70], (b) task [3] [59] [27] [72], and (c) others, however parallel/distributed algorithms are not discussed in this paper.

4.2.1 Complete Sequential Pure-Apriori Like ARM algorithms

Complete sequential pue-Apriori like ARM algorithms are exemplified by Apriori, AprioriTid and AprioriHybrid. The initial Apriori algorithm described a breadth-first (BFS) level-wise approach that employed a bottom-up approach to the generation of frequent item sets. Apriori-Tid was especially directed datasets where the number of possible candidate item sets (combinations) is less than the number of records in the data set:

$$\text{numberof combinatio} = 2^m - 1$$

AprioriTid also made use of the *apriori-gen* function (a key procedure of Apriori algorithm), but did not use database *D* for counting the supports after the first pass. AprioriHybrid is a combination of Apriori and AprioriTid. All three of these algorithms (Apriori, Apriori-TID and AprioriHybrid) required multiple database scans, *i.e.* significant disk I/O.

4.2.2 Partitioned Pure-Apriori Like ARM algorithms

Examples of the partitioned sequential pue-Apriori like ARM include MONET [31] and partition [56].

MONET was proposed by Holsheimer *et al* in 1995, it operates using the *vertical projection* of a given dataset (TID lists) and finds association rules by applying union an intersection to the TID lists. The vertical partition concept is also employed by Partition [56].

To reduce the number of database passes, the Partition algorithm [56] horizontally segments the data set into *n* phase/partitions. Partition is the first ARM algorithm to use horizontal partitioning. An ARM algorithm that divided the data set vertically was introduced in 2002 [66].

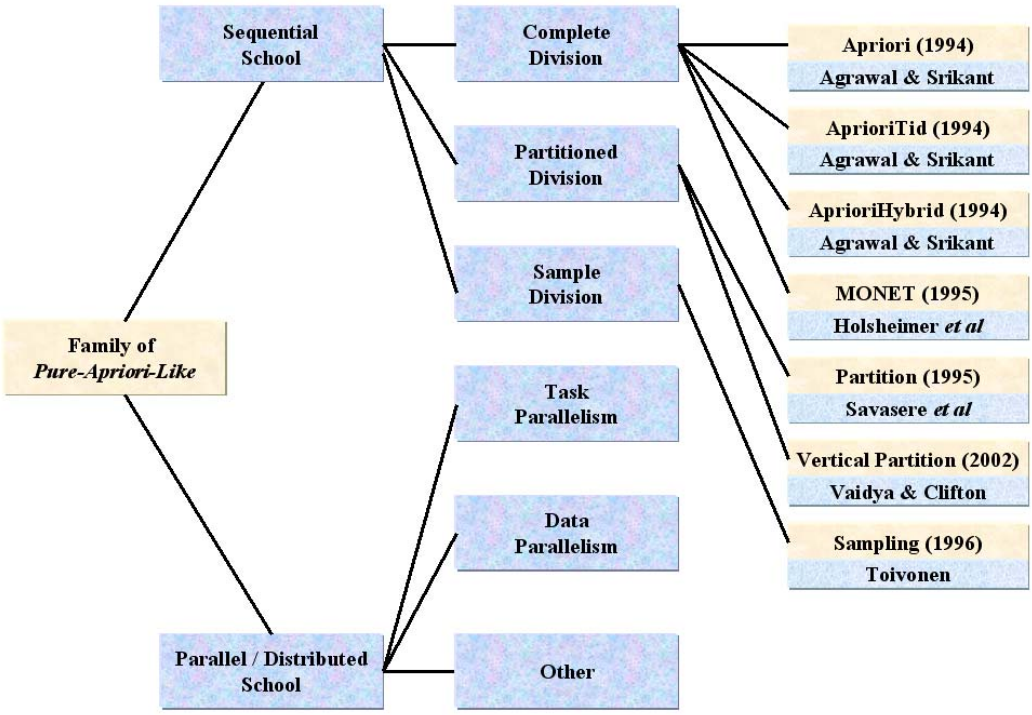

Figure 3: Family of Pure-Apriori-Like

4.2.3 Sampling Pure-Apriori Like ARM algorithms

Sampling [64], reduces the number of database scans to one in the "best" case and two in the "worst". The idea of Sampling is first introduced in [41], *"small samples are usually quite good for finding covering sets."* It was also suggested in [56] as a "future direction" – *"the problem of accurately estimating the number of partitions given the available memory, however, needs further work."* The idea of sampling is to first mine (using for example Apriori) a segment of the data (that will fit into primary storage) and identify candidate large item sets. To avoid omitting any possible candidate item sets sampling algorithms tend to make use of a "negative boarder" and a reduce support threshold. The negative boarder describes the minimal infrequent item sets as first described in [11].

The principal disadvantage associated with sampling algorithms is that they do not guarantee to find all frequent itemsets ---*"Partition can consider more but never fewer candidate itemets than Apriori"* as Roberto and Bayardo summarized in [53], *Pure-Apriori-Like*, from Apriori to Sampling, had made a significant contribution in the history of ARM, which the number of database scans had been deducted from *"m+1"* to lesser or equal to *"2"*.

4. 3 Family of *Dynamic-Like*

The concept of handling a non-complete database, as another ARM milestone, was initially raised by both Partition [56] and Sampling [64], which leads the appearance of the third category of ARM algorithms, *Dynamic-Like* (Figure 4).

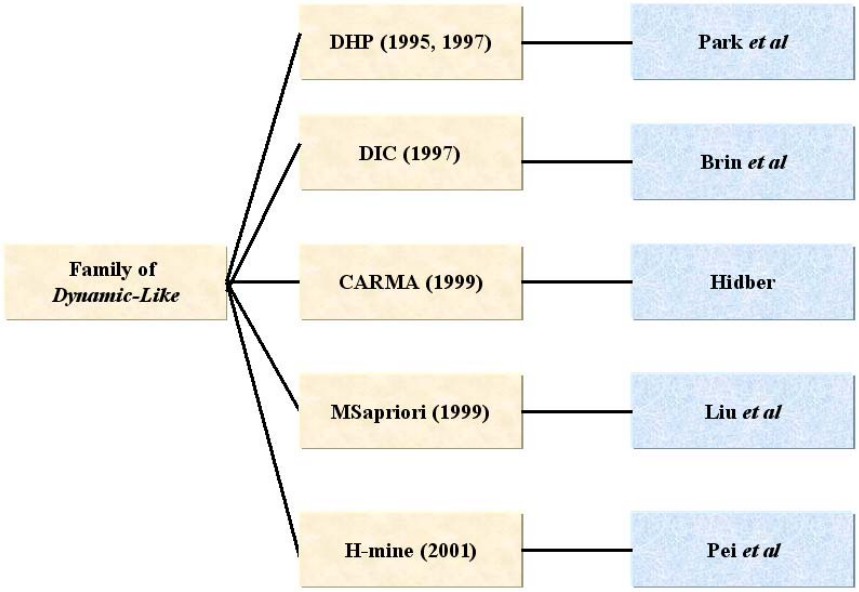

Figure 4: Family of Dynamic-Like

Brin *et al* in [11] abstractly introduced the algorithm of DIC, "Dynamic Itemset Counting", that *"works like a train over the data with stops at intervals M transactions apart;"* and *"the 'passengers' on the train are itemsets."* In fact, DIC horizontally segments the database in compartments / intervals *M*. It dynamically starts with counting the candidate 1-itemsets in the first compartment, and generates the candidate 2-itemsets at the end of the first compartment. When turning into the second compartment, DIC continue counts the candidate 1- and 2-itemsets,

and generates the candidate 3-itemsets at the end of the second interval, and so on. Once the flag / marker reaches the M^{th} compartment, all the candidate 1-itemsets have been counted. If there are further candidate itemsets, the marker will move back to the first compartment from the M^{th}. Because of the nature of dynamic, DIC is considered to solve another major constraint in ARM, counting the supports of candidate itemsets over the database in less time. However, the number of database scans, in general, is between *"2"* and *"m+1"*. We summarize the formula of the accurate passes calculation in DIC is *"1 + (m - 1) / M"*, where *m* represents the length of maximal large / frequent itemset, and *M* demonstrates the number of segments in data. Brin *et al* also introduced *lattice* as drawing *Figure 2* in [11]. Based on the idea of *negative border* in Sampling, *lattice* directs / guides to the fourth generation of ARM algorithms, *Maximal-Like*.

Theoretically, the first *Dynamic-like* algorithm should be traced back to DHP [44] [46], "Dynamic Hashing and Pruning", which actually attempts to minimize the set of candidate itemsets by reducing the size of candidate 2-itemsets. Roberto and Bayardo in [53] marked that DHP "uses the hashing scheme to re-write a smaller database after each pass in order to reduce the overhead of subsequent passes". Further, Park *et al* in [46] also combined Sampling to the framework of DHP for a higher accuracy of rules discovery.

In this family, another main person is named CARMA [30], "Continuous Association Rules Mining Algorithm". CARMA shows its dynamical feature by allowing users to easily change the thresholds of support and confidence at any transaction during the first database scan. It tries to bring the frequent itemsets computation on line, and scans the database exactly *"2"* times.

MSapriori [38] is a flexible and powerful model that *"allows users to specify multiple minimum item supports"*, and shows the dynamical ideology with setting of *"min_sup"*.

H-mine [50], as the most recent known algorithm with a dynamical thinking style, employs the H-struct, "hyperlinked data structure" and, which links *Dynamic-Like* to the fifth ARM algorithms generation, *Tree-Structure-Like*. H-mine shows its dynamical and tree-structured natures simultaneously when handling a dense dataset, and dynamically integrating the H-struct with the FP-tree-based mining. Since H-Mine does not mainly depend on FP-tree structure, we do not consider it as a *Tree-Structure-Like* algorithm in this paper.

In this family, even though the number of database scans is between *"2"* and *"m+1"*, *Dynamic-Like* algorithms contribute on solving the ARM constraints that other than the number of passes, and brining the ideology of dynamics into ARM.

4. 4 Family of *Maximal-Like*

Most ARM researches agree the fourth category of ARM algorithms, *Maximal-Like* (Figure 5) tasking on the discovery of Maximal Frequent Itemsets (MFIs), has started since four related algorithms, Eclat, MaxEclat, Clique and MaxClique, introduced with two clustering schemes, *equivalence classes* and *maximal uniform hypergraph cliques*, and three *lattice* traversals, bottom-up, top-down, and hybrid top-down / bottom-up in [71]. Both the Eclat-Like algorithms employ *equivalence classes*, while the Clique-Like uses *maximal uniform hypergraph cliques*. In another degree of view, Eclat and Clique work with bottom-up traversal, whereas the other two in Max-Like traverse the *lattice* by hybrid scheme.

In fact, a novel series of seven related *Maximal-Like* algorithms [25], *A_Random_MSS*, *All_MSS*, *A_Random_MFS*, *All_MFS*, *A_Random_MAS*, *A_Random_MAK*, and *All_MAK* was published a few months ahead of [71] that demonstrates a randomised approach even it does not guarantee to generate every MFI.

In early 1998, Lin and Kedem posted an algorithm building candidates and determining the Maximum Frequent Set (MFS) by hybrid top-down / bottom-up search approach, Pincer-Search

[36], which purposes to decrease the size of candidate set and reduce the number of database scans.

Concurrent to the work of Pincer-Search, Roberto and Bayardo introduced Max-Miner [53], an absolutely opposite algorithm from Apriori, directly applies the top-down search approach for mining the long patterns especially. Max-Miner has been used with some successes, but dependent on nature of data set, it works well if at least few maximal sets. This algorithm costs *"m+1"* passes, and another major disadvantage is that there is a requirement to continually obtain support counts for particular data sets therefore it tends to operate best using vertical data.

Dense-Miner [54], as an improvement that was developed from Max-Miner, imposes additional constraints on the rules being mined to reduce further the search space. Although Roberto *et al*, in [54], did not emphasize on Dense-Miner's usage in mining MFIs, we still consider that Dense-Miner as a member of *Maximal-Like*.

Two years later than Max-Miner, DepthProject [4], as the one attempting to bring the *depth-first* traversal (DFS), instead of BFS, to *lookaheads*, the key of Max-Miner, was created by Agarwal *et al*. DepthProject starts to represent the lexicography by tree data structure, and balance its children nodes distribution by dynamically reordering them. The appearance of DepthProject coincides to the appearance of the fifth generation in ARM algorithms, *Tree-Structure-Like*, which proves that ARM researches have begun to look into the tree-structured algorithms since around 1999.

In 2000, algorithms Closure, MaxClosure and AltMaxClosure were presented by Cristofor *et al* in [18], as the applications of *Galois connections*. In Details, Closure algorithm scans database *"(m/2)+1"*, whereas MaxClosure requires *"m-1"* passes in the worst case. In [18], Cristofor *et al* also attempted to signify the usage of closed itemset for finding all frequent itemsets by defining the term closure in itemset. Thus we conclude these three similar Apriori-based algorithms arrowhead to FCI. By exerting of three pruning strategies for the replacement of non-maximal itemsets,

MAFIA [12], "Maximal Frequent Itemset Algorithm", points to the concept of Vertical Mining again since using the vertical bit-vectors, and was introduced by Burdick *et al* in 2001. It should be paid attention here that Burdick *et al* clearly stated a straightforward relationship in the literature of MAFIA, *"MFI* $\subseteq$ *FCI* $\subseteq$ *FI"*, where *FI* signifies frequent itemset. This meaningful relationship adding up the stating of closure in [18] strongly demonstrates that *Maximal-Like* closely connects to the sixth generation of ARM algorithms, *Closed-Like*. Further, MAFIA should not only be thought as a *Maxiaml-Like* algorithm but also a member of *Closed-Like*.

GenMax [24], as the current best algorithm for mining MFIs, applies the idea of *backtrack* search combining the core of another novel Vertical Mining algorithm, *Diffset* [76]. GenMax requires a vertical *tidset / tidlist* format and exerts both the vertical and horizontal formats during the process of searching the MFIs.

In 2002, Yang *et al* presented a border-collapsing algorithm [69] for mining MFIs in a minimal number of database scans with a high degree of confidence in noisy environments.

Maximal-Like hardly works on the constraint in ARM, "minimizing the set of candidate itemsets", by avoiding candidate generation overhead, especially when the passing a database contains some long patterns already. Since the most of *Maximal-Like* algorithms employ or elect the vertical / inverted database layout involving the intersections of *tidsets*, primarily introduced in [71], algorithms in this family compute the support of itemsets simply and fast. Thus, we mark here that *Maximal-Like* is the first family of ARM algorithms that points to the conceptual approach of Vertical Mining. Even Zaki in [74] critically argued that mining MFIs causes many redundancies in rules generation and the problem of database scans, *Maximal-Like* algorithms have made a significant contribution to ARM by stating its principle, "every subset of MFI, just out of *negative*

border [64], is large / frequent and should be written in the *positive border / border of frequent itemsets* [71]."

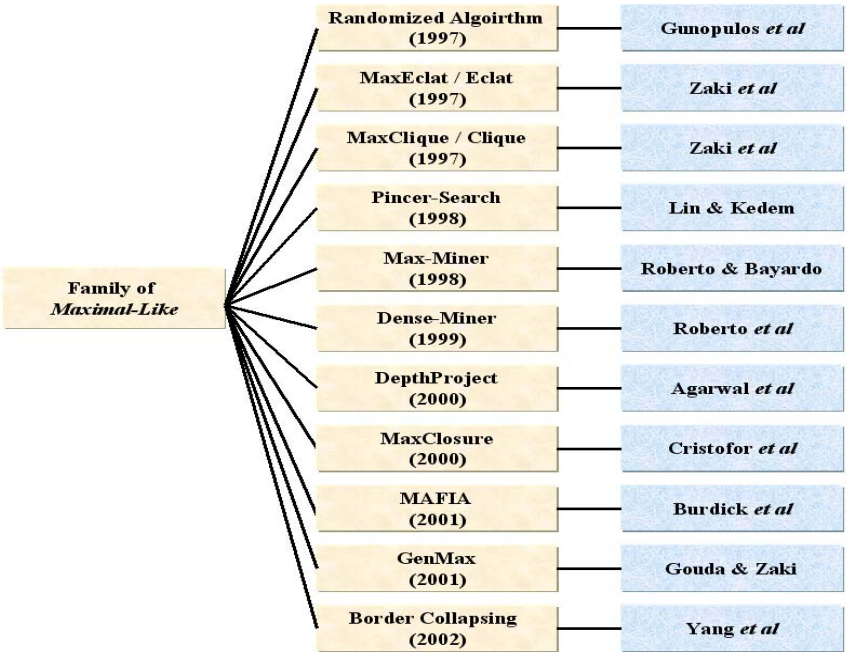

Figure 5: Family of Maximal-Like

4. 5 Family of *Tree-Structure-Like*

As we mentioned above, the fifth ARM algorithms category, *Tree-Structure-Like*, was born around 1999 that contains two main streams, school of Apriori-TFP [26] [15] [16] [17] [6] and school of FP-tree [28] [49] [57] [35] [29] [39] [67] [21] [14] (Figure 6).

The school of Apriori-TFP involves a series of several related algorithms, methods and ideas. Apriori-TFP was originally proposed with the structure of P-tree [26], "Partial Support Tree", aiming to decrease the size of candidate set. P-tree, in fact, stores the partial calculated supports to a set of *enumeration tree* and uses it for rules generating. In other words, P-tree is a reorganisation of the input data, into a "compressed" form, with the inclusion of partial totals.

Based on P-tree, the second tree structure in this school, "Total Support Tree" (T-tree), was developed, in [15], for support-counts storing. With the introduction of T-tree, the core of Apriori-TFP, algorithm TFP was proposed, which clearly demonstrates how to compute the total-supports from the partial-supports. In [16], the technique of itemsets ordering was pointed out, and it works well with Apriori-TFP: tree ordering by descending frequency of attributes improves the result of performance, especially when most common attributes are placed at the start of the set order.

Other than directly employing the standard Apriori algorithm by a hash tree data structure or generating T-tree right after generating P-tree, three innovative methods were introduced in [17]: generating sparse attributes T-tree with / without *X-check* for common sets that based on processing common attributes using *Brute-Force* algorithm; and generating sparse attributes T-tree and then finding support for common attribute sets that appear in T-tree using the second T-tree.

The most recent work on Apriori-TFP, called HSVP [6], is consisted by four creative ideas that hybridize TFP with three *Pure-Apriori-Like* algorithms, Partition, Sampling and Vertical Partition. Especially the method, *"TFP with Vertical Partitioning"* based on a horizontal database

segmentation, successfully reduces the database scan to *"1"*, and generates rules without candidates.

The second main stream has begun with an efficient FP-tree-based mining method, FP-growth [28], presented by Han *et al* in the same year as P-tree introduced. The same as Apriori-TFP, FP-tree-based algorithms seek rules without generating candidates either, and FP-growth algorithm, "Frequent Pattern Growth", only scans database *"twice"*. FP-growth represents transactions in a compressed tree structure, and stores the passing database by a mixed vertical / horizontal layout. It mines long patterns / MFIs by calling the approach of database projection recursively, which backwardly points to the generation of *Maxiaml-Like*. With DFS, FP-growth makes a great contribution to effectively associate different data structures comprising tree, linked list, and hash table. The actual data transactions are stored in a trie structure, a tree-based data structure for storing strings in order to support fast pattern matching, and each item has a linked list data structure to connect the next self / occurrence through all transactions. Also, a table is created to gather the separate itemsets with their supports, and each itemset points to the first occurrence in the trie structure.

In the same year as FP-growth proposed, an FP-tree-based database projection method, CLOSET [49], was proposed to deal with the closed itemsts. Thus, CLOSET is also considered as a *Closed-Like* algorithm that quickly mines FCIs by applying the structure of FP-tree.

By improving and innovating CLOSET, Pei and Han *et al* continually presented another two *Tree-Structure-Like* mixing *Closed-Like* algorithms, TFP [29] and CLOSET+ [67], for FP-tree-based FCI discovery.

In 2001, LPMiner [57], the first algorithm for finding the *length-decreasing support constraint* based frequent itemsets, was developed by Seno and Karypis. For the task of storing transactions / rows in main memory, LPMiner utilizes the FP-tree structure that was proposed with algorithm FP-growth.

A year later of LPMiner, Liu *et al* presented algorithm OP [39], "OpportuneProject". It combines DFS with BFS, and mines the complete frequent set with a bi- tree-based / array-based representation for projected transaction subsets. With presenting of the algorithm, Liu *et al* also introduced, in this paper, about the ideas of *bottom-up projection vs. top-down projection* and *pseudo projection vs. physical projection*.

The algorithm CATS Tree Builder and the algorithm FELINE [14], supported by CATS Tree data structure, an extension of FP-tree, allow finding all frequent itemsets without rebuilding the tree structure, and reduce the number of database scans to *"1"*. Thus, we conclude that even though the structure of FP-tree is considered complex and it requires a large amount of storage space, FP-tree-based methods efficiently and effectively return the set of frequent itemsets without generating candidates in a single pass.

Recently, Ei-Hajj and Zaiane proposed a disk-based and COFI-tree structured, "Co-Occurrence Frequent-Item-Tree", algorithm, Inverted Matrix [21]. With presenting database in vertical layout, this charming algorithm reduces the number of database scans, in some cases, less than *"1"*.

FP-tree not only contributes significantly on ARM, but also on Classification Rule Mining (CRM), another major KDD task / technique. The method CMAR [35], "Classification based on Multiple Association Rules", describes how FP-tree be used to create a structure of *class distribution-associate FP-tree*.

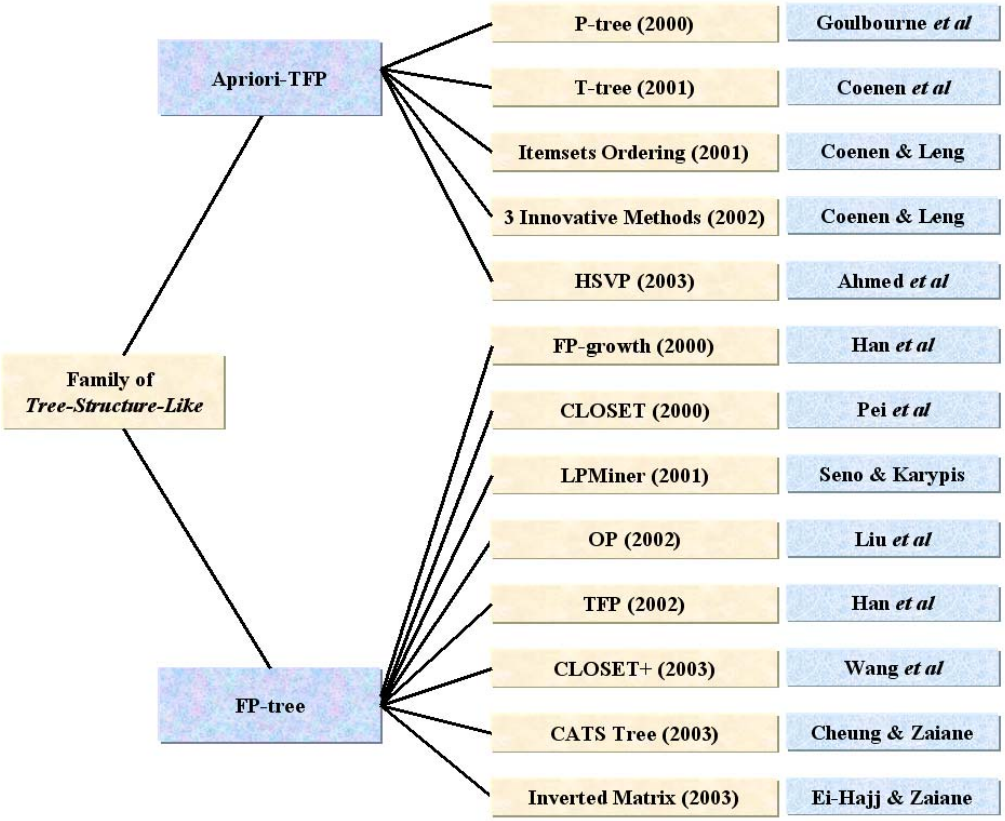

Figure 6: Family of Tree-Structure-Like

The algorithms in this family make a lot of efforts to reduce the numbers of database scans and candidates. Although FP-tree, CLOSET and CLOSET+ are definitely based on horizontal database representation, this family starts to consider about inverted database layout in purpose of the scalability and efficiency in both runtime and space usage. Thus, we signify that *Tree-Structure-Like* points to the approach of Vertical Mining as *Maximal-Like* does.

4. 6 Family of *Closed-Like*

Another recent promising direction in ARM algorithms focuses on finding FCIs, which is extremely prosperous and flourishing that directs to the sixth generation of ARM algorithms, *Closed-Like* (Figure 7). We have just mentioned twice about *Closed-Like* when introducing the algorithm MAFIA in *Maximal-Like* and the algorithms CLOST, TFP and CLOSET+ in *Tree-Structure-Like*. The principle of *Closed-Like*, based on the one of *Maximal-Like*, should be summarized as, "all subsets of a given FCI are frequent since FCI is a superset of MFI; and FCIs can be used for rule generation even MFI cannot".

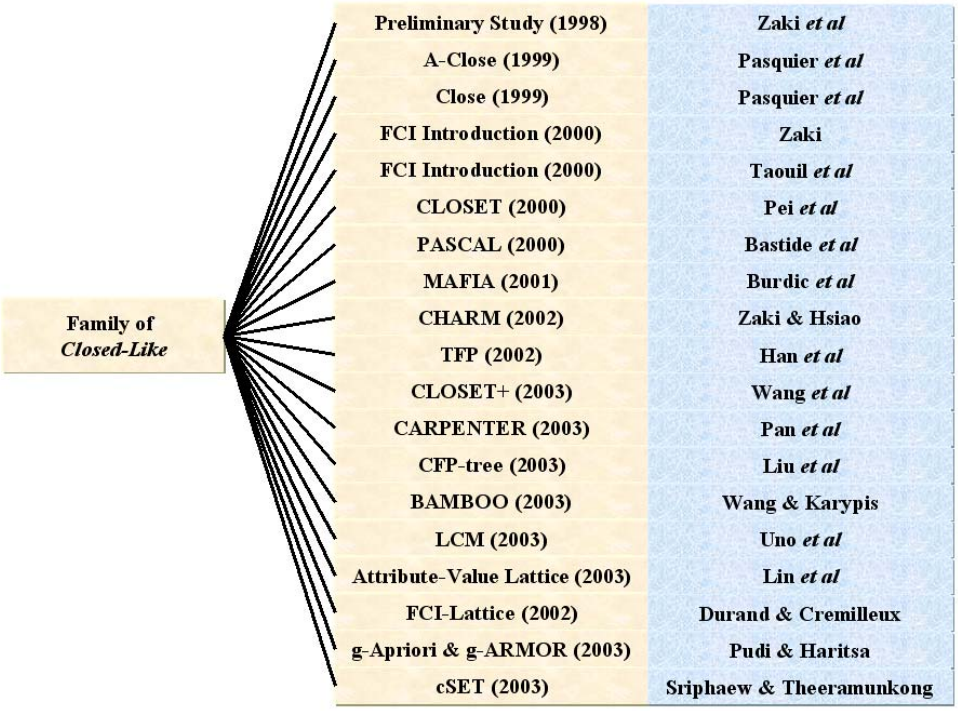

Figure 7: Family of Closed-Like

The framework of closed itemset has been formally and fully introduced with its definition, lemmas and theorems in two papers [63] [74] in 2000. Defining this framework, *"a closed itemset as an itemset X that is that same as its closure"* [74], with the term frequent aiming to *"constitute a minimal non-redundant generating set for all frequent itemsets and their support, and thus for all association rules, their support and their confidence"* [63]. The initial model and the preliminary of FCI, arguably, should be tracked back to [73], with showing of a lattice-theoretic framework for finding a small rule generating set.

In 1999, Pasquier *et al* firstly addressed two *Closed-Like* algorithms, A-Close [47] and Close [48], and determined the relationship between FCI and MFI: MFIs are maximal FCIs. A-Close, an Apriori-based FCI mining algorithm, uses bottom-up search approach to identify the smallest frequent itemsets for determining the closed itemsets, and then calculates the closures for each found smallest frequent itemset. Close returns FCIs by pruning the *closed itemset lattice*, a suborder of the *subset lattice* in formal concept analysis.

Concurrently, Zaki and Hsiao designed algorithm CHARM [75], structured by mixing a tree approach, *dual itemset-tidset tree*, and a fast hash-based approach, and was tardily published in 2002. CHARM initially stores the closed itemsets in a *dual itemset-tidset tree* with showing both the itemset space and transaction space simultaneously. By a hybrid BFS / DFS and a vertical data representation, *diffset* [76], it quickly counts frequencies and fast finds all FCIs in the second stage. Finally, it employs the hash-based approach to replace the "non-closed" itemsets. We flag here that CHARM, as an absolute *Closed-Like* algorithm, also points to the family of *Tree-Structure-Like* and the term of Vertical Mining that we shortly described above, since the assisting from *dual itemset-tidset tree* and *diffset* [76].

In May 2000, CLOSET algorithm [49] was posted, as we mentioned above, it tightly connects *Tree-Structure-Like* and *Closed-Like*.

PASCAL algorithm is an optimization of Apriori, published in [8], using for FCIs discovery with requiring *"2"* passes. Since it works based on *pattern counting inference*, seeking and returning

of the *frequent key patterns*, and the definition of *frequent key pattern* identifies FCI's, we group it in the family of *Closed-Like*.

In 2001, MAFIA [12], as we alluded above, was born with mining both MFIs and FCIs that belongs to both families of *Maximal-* and *Closed-Like*.

TFP [29], as another hybrid *Tree-Structure-* and *Closed-Like* algorithm, concentrates on mining top-*k* FCIs with a length threshold, *min_l*, and was presented by Han *et al* in 2002. The two new parameters, *k* and *min_l*, represent a user-desired number of FCIs being discovered, and a number of the minimal length of each FCI. Instead of being benchmarked by support threshold, TFP addresses the length threshold tactics for mining FCIs would efficiently fix the constraint in ARM algorithms that other than reducing the number of passes or candidates: counting the supports in less time. In addition, TFP exerts a hybrid top-down / bottom-up FP-tree mining strategy to return FCIs speedily.

In August 2003, the most updated FP-tree-based FCI mining algorithm, CLOSET+ [67], was proposed by Wang *et al*. In comparison to FP-tree and CLOSET, CLOSET+ not only utilizes a bottom-up *tree-projection*, but also a top-down one. For quickly growing FCIs in different database aspects, *dense vs. parse*, it actually unites together a bottom-up *physical tree-projection* and a top-down *pseudo tree-projection*.

Concurrently and independently, another two novel FCI mining algorithms CARPENTER [43] and CFP-tree [40] were introduced in the same KDD conference as CLOSET+, *SIGKDD'03*. CARPENTER is an algorithm dealing with a bioinformatics dataset, having a large number of items and a small number of transactions. It performs a DFS over the *row set enumeration tree*, a tree structure vertically listing of all transaction-combinations by bottom-up approach, to find FCIs through three related pruning steps. CFP-tree, "Condensed Frequent Pattern Tree", is a disk-based data structure being used to store and query frequent itemsets on disk.

One month later than CLOSET+, CARPENTER and CFP-tree, another FCI mining algorithm, BAMBOO [68], was developed by Wang and Karypis. This algorithm attempts to bring the concept of *length-decreasing support constraint* into enumerating FCIs.

In 2003, there were another three similar FCI seeking algorithms, LCMfreq, LCM and LCMmax, introduced in [65]. Uno *et al*, in [65], focused on the development of LCM, "Liner time Closed item set Miner", since both LCMfreq and LCMmax are the variations of LCM for finding all frequent itemsets and MFIs. The key to effectively control LCM's computation time is that LCM could intelligently employ either *occurrence deliver* with *right-first sweep* or *diffsets* [76].

Since Zaki *et al* firstly represented a full set of all possible item-combinations / itemsets by *lattice* in [71], it has become an indispensable graphing technique in ARM, especially in generating FCIs. For instance, attribute-value lattice is used to find FCIs, which elects a vertical database representation. Lin *et al* proposed this latticed method, in [37], which initially constructs an attribute-value lattice over the database, and then traverses the lattice by bottom-up approach and DFS approach and finally discovers all FCIs. An FCI-Lattices could also be used to mine the clusters from categorical data [20]. This idea connects ARM to another important KDD task, Clustering.

Some ARM researchers start to look into finding of the generalized FCIs [51] [62]. In [51], two algorithms were demonstrated by Pudi and Haritsa, g-Apriori and g-ARMOR, which based on the idea of integrating generalized itemsets into the concept of FCI. In fact, generalized closed itemset, as a new framework, contains two properties: *"the supports of all frequent itemsets can be derived from the output within an error of ε; and if $\varepsilon = 0$, the output is precisely the frequent closed itemsets"*.

In [62], Sriphaew and Theeramunkong introduced algorithm cSET for growing Generalized Association Rules [61] [60], another extension from the traditional ARM by using hierarchical

taxonomy over the items in database. The concept of this framework is that a large set of generalized itemsets always can be represented by a smaller set of generalized closed itemsets. Thus, cSET optimally emphasizes on enumerating only a small set of generalized FCIs following some constrains and conditional properties.

The family of *Closed-Like* is thriving that contains many members, and some of them link to different sub-fields in ARM, such as Clustering ARM and Generalized ARM. Also this family closed connects another two ARM algorithms families, *Maximal-* and *Tree-Structure-Like*, that points to the term, Vertical Mining, in ARM. Mining FCIs avoids the redundancies in finding all frequent association rules that would effectively management the computational time. Since a number of *Closed-Like* algorithms are also *Maximal-* or *Tree-Structure*-based, we confidently conclude that the generation of *Closed-Like* inheriting the properties from *Maximal-* and *Tree-Structure-Like* generations that does well of reducing the numbers of passes and candidates, and rationally controls the costs of counting supports over the database with mining rules non-redundantly.

4. 7 Border of Vertical Mining

Since the fourth, fifth and sixth generations in ARM algorithms strongly relate and point to each other in both directions, and the most members of them require / elect an inverted database representation, we summarize that they all belong to the conceptual approach of Vertical Mining, or we say that all the three generations in ARM algorithms are Vertical-based generations / families. Other related Vertical Mining algorithms, but excluded from any summarized families, include VIPER [58], SPAM [7], Diffset [76], and Medic [23] (Figure 8).

VIPER [58] is actually the abbreviation of "Vertical Itemset Partitioning for Efficient Rule-extrction", presented by Shenoy *et al* in 2000. Algorithm VIPER scans database in *"2"* passes: it, first of all, counts all frequent 1-itemsets and re-constructs the them from a classical horizontal data layout to *1-snakes*, a compressed bit-vector itemsets representation with frequent 1-itemsets; then it computes the frequent 2-itemsets and returns a frequent set through subsequent passes.

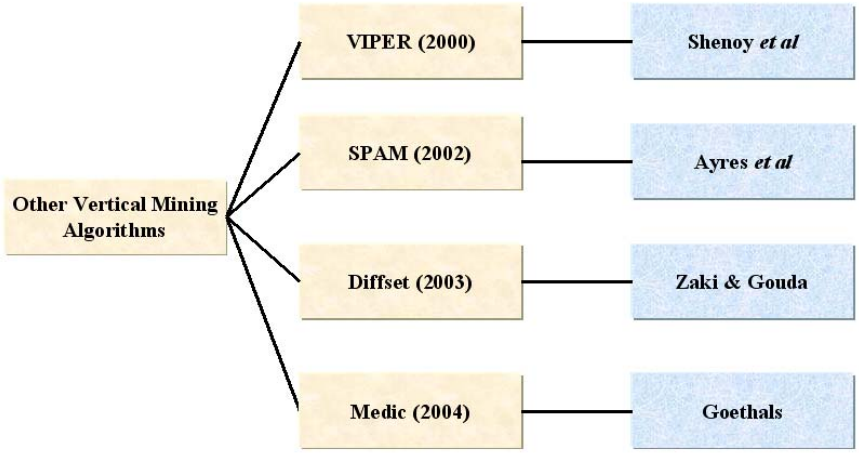

Figure 8: Related Vertical Mining algorithms

SPAM [7], such as MAFIA [12] in *Maximal-Like* and VIPER [58], also employs vertical bit-vectors for fast itemset and sequence generating, by Ayres *et al* in 2002. SPAM, "Sequential Pattern Mining", is the first sequential pattern mining algorithm that uses the strategy of DFS.

As we described above, some ARM algorithms utilize the opinion of *diffset* [76] comprising GenMax [24], CHARM [76] and LCM [65]. Diffset is a vertical data representation, which focuses on tracking of differences in the *tidlists* of a candidate pattern from its generating frequent patterns, and works well on cutting down the size of memory of storing the intermediate results.

Medic [23], as the most updated algorithm in ARM is going to be presented by Goethals in March 2004. In [Geothals 2004], Goethals detailedly depicted Eclat [71] and FP-growth [28] in comparisons and proposed algorithm Medic at the end, which saves the main memory by not maintaining the covers / *tidlists* of the past itemsets.

5. CONCLUSION

In this paper, we reviewed the principle of ARM with summarizing the three major frameworks in measuring the "interestingness" of association rules. We presented a structure to categorize the most common existing ARM algorithms with their variant characters. We clearly show both the internal and external relationships among the six different categories in ARM algorithms and extensions, and other related KDD topics in Figure 9.

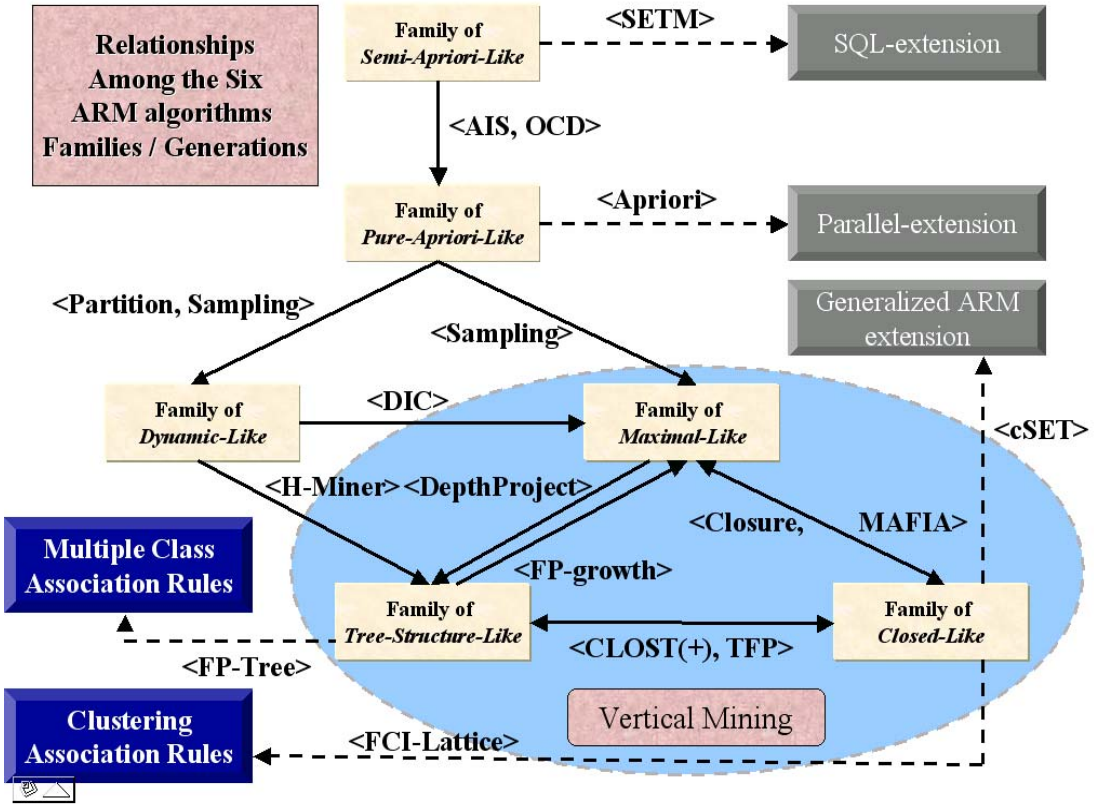

Figure 9: Relationships among the six categories and the related topics

From *Semi-Apriori-Like* generation to *Closed-Like* generation, ARM algorithms have passed a full decade, 1993-2004, and started to move into the conceptual approach of Vertical Mining from the traditional horizontal representation approach. The number of database scans has been successfully reduced to *"1"* or, even for some cases, less than *"1"*. By turning into *Tree-Structure-Like*, ARM algorithms have begun to enumerate the frequent set without generating candidates. The family of *Maximal-Like* efforts on decreasing the time of counting supports, and

also on mining the long patterns by top-down search approach instead of the traditional bottom-up one. *Closed-Like* algorithms succefully contribute on finding frequent rules without redundancies. During summarizing the existing ARM algorithms, we remark six important and meaningful issues that would useful and helpful for the future ARM researches: horizontal layout vs. vertical layout; bottom-up approach vs. top-down approach; BFS vs. DFS; tree data structure vs. other data structures; physical projection vs. psudo projection; and FP-tree vs. diffset in data compression techniques.

References

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large database. In *Proceedings of the 1993 ACM SIGMOD Conference*, pages 207-216, 1993.
[2] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules. In *Proceedings of the 20th VLDB Conference*, Santiago, Chile, 1994.
[3] R. Agrawal and J. C. Shafer. Parallel Mining of Association Rules. In *Proceedings of IEEE Transactions on Knowledge and Data Engineering, Vol. 8, No. 6,* pages 962-969, December 1996.
[4] R. C. Agarwal, C. C. Aggarwal and V. V. V. Prasad. Depth First Generation of Long Patterns. In *Proceedings of KDD 2000,* Boston, MA, USA.
[5] C. C. Aggrawal and P. S. Yu. Mining large itemsets for association rules. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering,* 21(1): 23-31, March 1998.
[6] S. Ahmed, F. Coenen and P. Leng. Strategies for Partitioning Data in Association Rule Mining. In *Proceedings of AI2003, the 23rd SGAI Intl. Conf. on Innovative Techniques and Applications of Artificial Intelligence,* pages 127-139, Cambridge, U.K., December 2003.
[7] J. Ayres, J. E. Gehrke, T. Yiu and J. Flannick. Sequential Pattern Mining Using Bitmaps. In *Proceedings of SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining,* July 2002.
[8] Y. Bastide, R. Taouil, N. Pasquier, G. Stumme and L. Lakhal. Mining Frequent Patterns with Counting Inference. In *Proceedings of SIGKDD Explorations, 2(2),* December 2000.
[9] M. J. Berry and G. S. Linoff. Data Mining Techniques for Marketing, Sales and Customer Support. John Wiley and sons, 1997
[10] F. Berzal, I. Blanco, D. Sanchez, and M.A. Vila. Measuring the accuracy and interest of association rules: A new framework. *Intelligent Data Analysis,* submitted, 2002.
[11] S. Brin, R. Motwani, J. D. Ullman, S. Tsur. Dynamic Itemset Counting and Implication Rules for Market Basket Data. In *Proceedings of the ACM SIGMOD Conference*, pp. 255-264, 1997.
[12] D. Burdick, M. Calimlim and J. Gehrke. MAFIA: A Maximal Frequent Itemset Algorithm for Transactional Databases. In *Proceedings of Intl. Conf. on Data Engineering,* April 2001.
[13] D. W-L. Cheung, J. Han, V. Ng, A. W-C. Fu and Y. Fu. A Fast Distribute Algorithm for Mining Association Rules. In *Proceedings of PDIS,* 1996.
[14] W. Cheung and O. R. Zaiane. Incremental Mining of Frequent Patterns Without Candidate Generation or Support Constraint. In *Proceedings of IDEAS 2003,* Hong Kong.
[15] F. Coenen, G. Goulbourne and P. Leng. Computing Association Rules Using Partial Totals. In *Proceedings of PKDD2001, eds L. De Readt and A Siebes, LNAI 2168,* pages 54-66, August 2001.
[16] F. Coenen and P. Leng. Optimising Association Rule Algorithms Using Itemset Ordering. *Research and Development in Intelligent Systems XVIII: Proc ES2001 Conference, eds M Bramer, F Coenen and A Preece, Springer,* pages 53-66, 2001
[17] F. Coenen and P. Leng. Finding Association Rules with Some Very Frequent Attributes. In *Proceedings of PKDD2002, eds T. Elomaa et al, LNAI 2431,* pages 99-111, 2002
[18] D. Cristofor, L. Cristofor and D. Simovici. Galois Connection and Data Mining. In *Proceedings of Journal of Universal Computer Science,* 6(1): 60-73, 2000.
[19] M. Delgado, M.J. Martin-Bautista, D. Sanchez, and M.A. Vila. Mining Text Data: Special Features and Patterns. Pattern Detection and Discovery, pages 140-153. Springer, 2002. In *Proceedings of the ESF Exploratory Workshop, LNAI 2447*, London, U.K., September 2002.
[20] N. Durand and B. Cremilleux. Extration of a Subset of Concepts from the Frequent Closed Itemset Lattice: A New Approach of Meaningful Clusters Discovery. In *Proceedings of the International Workshop on Advances in Formal Concept Analysis for Knowledge Discovery in Databases (FCAKDD), Co-located with the 15th European Conference on Artificial Intelligence (ECAI'02),* pages 24-25, Lyon, France, July 2002.

[21] M. Ei-Hajj and O. R. Zaiane. Inverted Matrix: Efficient Discovery of Frequent Items in Large Datasets in the Context of Interactive Mining. In *Proceedings of SIGKDD'03,* Washington, DC, USA. August 24-27, 2003.
[22] Alex A. Freitas. Data Mining and Knowledge Discovery with Evolutionary Algorithms. *Springer-Verlag Berlin Heidelberg,* Germany, 2002.
[23] B. Goethals. Memory Issues in Frequent Itemset Mining. In *Proceedings of SAC'04,* Nicosia, Cyprus, March 14-17, 2004.
[24] K. Gouda and M. J. Zaki. Efficiently Mining Maximal Frequent Itemsets. In *Proceedings of the 1st IEEE Intl. Conf. on Data Mining,* November 2001.
[25] G. Gunopulos, H. Mannila and S. Saluja. Discovering All Most Specific Sentences by Randomized Algorithms. In *Proceedings of the 6th Intl. Conf. on Database Theory,* pages 215-229.
[26] G. Goulbourne, F. Coenen and P. Leng. Algorithms for Computing Association Rules using a Partial-Support Tree. In *J. Knowledge-Based Systems 13,* pages 141-149, 2000.
[27] E-H. Han, G. Karypis and V. Kumar. Scalable Parallel Data Mining For Association Rules. In *Proceedings of the ACM SIGMOD Conference, pages 277-288,* 1997.
[28] J. Han, J. Pei and Y. Yin. Mining Frequent Patterns Without Candidate Generation. In *Proceedings of SIGMOD'00,* Dallas, TX, May 2000.
[29] J. Han, J. Wang, Y. Lu and P. Tzvetkov. Mining Top-k Frequent Closed Patterns Without Minimum Support. In *Proceedings of 2002 IEEE Intl. Conf. On Data Mining (ICDM'02),* Maebashi City, Japan. December 09-12, 2002.
[30] C. Hidber. Online Association Rule Mining. In *Proceedings of ACM SIGMOD Intl. Conf. on Management of Data (SIG-MOD'99)*, Philadelphia, PA, USA, May-June 1999.
[31] M. Holsheimer, M. Kersten, H. Mannila and H. Toivonen. A Perspective on Databases and Data Mining. In *Proceedings of 1st Intl. Conf. on Knowledge Discovery and Data Mining (KDD'95),* pages 150-155, Montreal, Canada, August 1995.
[32] M. Houtsma and A. Swami. Set-oriented Mining of Association Rules. Research Report RJ 9567, IBM Almaden Research Center, San Jose, California, October 1993.
[33] M. Houtsma and A. Swami. Set-oriented Mining of Association Rules in Relational Databases. In *Proceedings of 11th Intl. Conf. on Data Engineering (ICDE'95)*, Taipei, Taiwan, March 1995.
[34] M. A. W. Houtsma and A. Swami. Set-oriented Mining in Relational databases, *Data and Knowledge Engineering*, 1996.
[35] W. Li, J. Han and J. Pei. CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules. In *Proceedings of 2001 IEEE Intl. Conf. on Data Mining (ICDM 2001),* pages 369-376. IEEE Computer Society Press, 2001.
[36] D-I. Lin and Z. M. Kedem. Pincer-Search: A New Algorithm for Discovering the Maximum Frequent Set. In *Proceedings of the 6th Intl. Conf. Extending Database Technology,* March 1998.
[37] T. Y. Lin, X. T. Hu and E. Louie. Using Attribute Value Lattice to Find Closed Frequent Itemsets. In *Proceedings of the 2003 SPIE Intl. Conf. on Data Mining and Knowledge Discovery Tools, Methods and Techniques,* Orlando, FL, USA. April 21-24, 2003.
[38] B. Liu, W. Hsu and Y. Ma. Mining Association Rules with Multiple Minimum Supports. In *Proceedings of KDD-99,* San Diego, CA, USA.
[39] J. Liu, Y. Pan, K. Wang, and J. Han. Mining Frequent Item Sets by Opportunistic Projection. In *Proceedings of SIGKDD'02,* July 2002.
[40] G. Liu, H. Lu, W. Lou and J. X. Yu. On Computing, Storing and Querying Frequent Patterns. In *Proceedings of SIGKDD'03,* Washington, DC, USA. August 24-27, 2003.
[41] H. Mannila, H. Toivonen and A. I. Verkamo. Efficient Algorithms for Discovering Association Rules. In *Proceedings of the AIII Workshop on Knowledge Discovery in Databases (KDD-94),* pages 181-192, July 1994.
[42] R. Meo, G. Psaila and S. Ceri. A New SQL-Like Operator for Mining Association Rules. In *Proceedings of the 22nd International Conference on Very Large Databases,* pp.122-133, Mumbai, India, 1996.
[43] F. Pan, G. Cong, A. K. H. Tung, J. Yang and M. J. Zaki. CARPENTER: Finding closed Patterns in Long Biological Datasets. In *Proceedings of SIGKDD'03,* Washington, DC, USA. August 24-27, 2003.
[44] J. S. Park, M-S. Chen and P. S. Yu. An Effective Hash Based Algorithm for Mining Association Rules. In *Proceedings of ACM SIGMOD Intl. Conf. on Management of Data (SIG-MOD'95),* pages 175-186, San Jose, California, USA, May 1995.

[45] J. S. Park, M-S. Chen and P. S. Yu. Efficient Parallel Data Mining for Association Rules. In *Proceedings of 4th Intl. Conf. on Information and Knowledge Management (CIKM'95),* pages 31-36, Baltimore, Maryland, USA, November 1995.
[46] J. S. Park, M-S. Chen and P. S. Yu. Mining Association Rules with Adjustable Accuracy. In *Proceedings of 6th Intl. Conf. on Information and Knowledge Management (CIKM'97),* pages 151-160, Las Vegas, Nevada, USA, November 1997.
[47] N. Pasquier, Y. Bastide, R. Taouil and L. Lakhal. Discovering Frequent Closed Itemsets for Association Rules. In *Proceedings of the 7th Intl. Conf. on Database Theory,* January 1999.
[48] N. Pasquier, Y. Bastide, R. Taouil and L. Lakhal. Efficient Mining of Association Rules Using Closed Itemset Lattices. In *Proceedings of Information Systems,* 24(1): 25-46, 1999.
[49] J. Pei, J. Han and R. Mao. CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets. In *Proceedings of SIGMOD Intl. Workshop on Data Mining and Knowledge Discovery,* May 2000.
[50] J. Pei, J. Han, H. Lu, S. Nishio, S. Tang and D. Yang. H-Mine: Hyper-Structure Mining of Frequent Patterns in Large Databases. In *Proceedings of ICDM'01,* November 2001.
[51] V. Pudi and J. R. Haritsa. Generalized Closed Itemsets for Association Rule Mining. In *Proceedings of 19th IEEE Intl. Conf. on Data Engineering (ICDE),* pages 714-716, Bangalore, India, March 2003.
[52] R. Rantzau. Processing Frequent Itemset Discovery Queries by Division and Set containment Join Operators. In *Proceedings of DMKD'03,* pages 20-27, San Diego, CA, USA. June 13, 2003.
[53] J. Roberto and Jr. Bayardo. Efficiently Mining Long Patterns from Databases. In *Proceedings of ACM-SIGMOD Int'l Conf. On Management of Data,* pages 85-93, 1998.
[54] J. Roberto, Jr. Bayardo, R. Agrawal and D. Gunopulos. Constraint-Based Rule Mining in Large, Dense Databases. In *Proceedings of the 15th Intl. Conf. on Data Engineering,* pages 188-197, 1999.
[55] M. Salim and X. Yao. Evolving SQL Queries for Data Mining. In *Proceedings of IDEAL 2002,* pages 62-67, 2002
[56] A. Savasere, E. Omiecinski and S. Navathe. An Efficient Algorithm for Mining Association Rules in Large Databases. In *Proceedings of the 21st VLDB Conference*, Zurich, Swizerland, 1995.
[57] M. Seno and G. Karypis. LPMiner: An Algorithm for Finding Frequent Itemsets Using Length-Decreasing Support Constraint. In *Proceedings of ICDM'01,* November 2001.
[58] P. Shenoy, G. Bhalotia, J. R. Haritsa, M. Bawa, S. Sudarshan and D. Shah. Turbo-charging Vertical Mining of Large Databases. In *Proceedings of Intl. Conf. Management of Data,* May 2000.
[59] T. Shintani and M. Kitsuregawa. Hash Based Parallel Algorithms for Mining Association Rules. In *Proceedings of PDIS,* 1996.
[60] T. Shintani and M. Kitsuregawa. Parallel Mining Algorithms for Generalized Association Rules with Classification Hierarchy. In *Proceedings of ACM SIGMOD Intl. Conf. on Management of Data (SIGMOD 1998),* pages 25-36, Seattle, Washington, USA. June 1998.
[61] R Srikant and R. Agrawal. Mining Generalized Association Rules. In *Proceedings of the 21st Intl. Conf. on Very Large Databases,* pages 407-419, Zurich, Switzerland, 1995.
[62] K. Sriphaew and T. Theeramunkong. Mining Generalized Closed Frequent Itemsets of Generalized Association Rules. In *Proceedings of KES2003, LNAI2773,* pages 476-484, 2003.
[63] R. Taouil, N. Pasquier, Y. Bastide and L. Lakhal. Mining Bases for Association Rules Using Closed Sets. In *Proceedings of the 16th Intl. Conf. on Data Engineering,* San Diego, California, Februery 28 – March 03, 2000.
[64] H. Toivonen. Sampling Large Databases for Association Rules. In *Proceedings of 22nd VLDB Conference*, Mumbai (Bombay), India, 1996.
[65] T. Uno, T. Asai, Y. Uchida and H. Arimura. LCM: An Efficient Algorithm for Enumerating Frequent Closed Item Sets. In *Proceedings of Workshop on Frequent itemset Mining Implementations (FIMI'03),* 2003.
[66] J. Vaidya and C. Clifton. Privacy Preserving Association Rule Minng in Vertically Partitioned Data. In *Proceedings of SIGKDD'02,* Edmonton, Alberta, Canada.
[67] J. Wang, J. Han and J. Pei. CLOSET+: Searching for the Best Strategies for Mining Frequent Closed Itemsets. In *Proceedings of SIGKDD'03,* Washington, DC, USA. August 24-27, 2003.
[68] J. Wang and G. Karypis. BAMBOO: Accelerating Closed Itemset Mining by Deeply Pushing the Length-Decreasing Support Constraint. DTC Research Report 2003/01, September 2003.
[69] J. Yang, W. Wang, P. S. Yu and J. Han. Mining Long Sequential Patterns in a Noisy Environment. In *Proceedings of ACM SIGMOD '2002,* Madison, Wisconsin, USA. June. 4-6, 2002
[70] M. J. Zaki, M. Ogihara, S. Parthasarathy and W. Li. Parallel Data Mining for Association Rules on Shared-Memory Multiprocessors. *Technical Report TR 618, University of Rochester, Computer Science Department,* May 1996.

[71] M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New Algorithms for Fast Discovery of Association Rules. In *Proceedings of the 3rd Intl. Conf. on Knowledge Discovery and Data Mining (KDD'97),* pages 283-286, Newport Beach, California, USA, August 1997.
[72] M. J. Zaki, S. Parthasarathy, M. Ogihara and W. Li. New Parallel Algorithms for Fast Discovery of Association Rules, In *Proceedings of Data Mining and Knowledge Discovery, Vol. 1, No. 4,* pages 343-373, December 1997.
[73] M. J. Zaki and M. Ogihara. Theoretical foundations of association rules. In *Proceedings of the 3rd ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery,* June 1998.
[74] M. J. Zaki. Generating Non-Redundant Association Rules. In *Proceedings of KDD 2000,* Boston, MA, USA.
[75] M. J. Zaki and C-J. Hsiao. CHARM: An Efficient Algorithm for Closed Itemset Mining. In *Proceedings of the 2nd SIAM Intl. Conf. on Data Mining,* April 2002.
[76] M. J. Zaki and K. Gouda. Fast Vertical Mining Using Diffsets. In *Proceedings of SIGKDD'03,* Washington, DC, USA. August 24-27, 2003.