Association Mining

AARON CEGLAR AND JOHN F. RODDICK

Flinders University of South Australia

The task of finding correlations between items in a dataset, association mining, has received considerable attention over the last decade. This article presents a survey of association mining fundamentals, detailing the evolution of association mining algorithms from the seminal to the state-of-the-art. This survey focuses on the fundamental principles of association mining, that is, itemset identification, rule generation, and their generic optimizations.

Categories and Subject Descriptors: H.3.3 [**Information Storage and Retrieval**]: Information search and Retrieval—*Retrieval models*; H.2.4 [**Database Management**]: Systems

General Terms: Algorithms

Additional Key Words and Phrases: Data mining, association mining

1. INTRODUCTION

Association (rule) mining, the task of finding correlations between items in a dataset, has received considerable attention, particularly since the publication of the AIS and Apriori algorithms [Agrawal et al. 1993; Agrawal and Srikant 1994]. Initial research was largely motivated by the analysis of market basket data, the results of which allowed companies to more fully understand purchasing behavior and, as a result, better target market audiences. For example, an insurance company, by finding a strong correlation between two policies A and B, of the form $A \Rightarrow B$, indicating that customers that held policy A were also likely to hold policy B, could more efficiently target the marketing of policy B through marketing to those clients that held policy A but not B. In effect, the rule represents knowledge about purchasing behavior. Association mining applications have since been applied to many different domains including market basket and risk analysis in commercial environments, epidemiology, clinical medicine, fluid dynamics, astrophysics, crime prevention, and counter-terrorism—all areas in which the relationship between objects can provide useful knowledge.

Association mining is user-centric as the objective is the elicitation of useful (or interesting) rules from which new knowledge can be derived. The key characteristics of *usefulness* suggested in the literature are that the rules are novel, externally significant, unexpected, nontrivial, and actionable [Bayardo Jr and Agrawal 1999; Dong and Li 1998; Freitas 1999; Hilderman and Hamilton 1999, 2001; Roddick and Rice 2001;

Authors' address: School of informatics and Engineering, Flinders University of South Australia, P. O. Box 2300, Adelaide 5003, South Australia; email: ceglar@infoeng.flinders.edu.au.

Sahar 1999; Silberschatz and Tuzhilin 1995, 1996]. The association mining system's role in this process is to facilitate the discovery, heuristically filter, and enable the presentation of these inferences or rules for subsequent interpretation by the user to determine their usefulness.

A decade on from the seminal work of Agrawal et al. [1993] association mining analysis has become a mature field of research. The fundamentals of association mining are now well established and there appears little current research on optimizing the performance of classic itemset identification. Notable exceptions to this include the adaptation of Inductive Logic Programming (ILP) techniques to the field of association mining [Deshaspe and Toivonen 1998] and the development of parallel and distributed variations of established itemset identification algorithms [Zaki 1999]. The majority of current research involves the specialization of fundamental association mining algorithms to address specific issues, such as the development of incremental algorithms to facilitate dynamic dataset mining or the inclusion of additional semantics (such as time, space, ontologies, etc.) to discover, for example, temporal or spatial association rules.

Association mining analysis is a two part process. First, the identification of sets of *items* or *itemsets* within the dataset. Second, the subsequent derivation of *inferences* from these itemsets. The majority of related research to date has focused upon the efficient discovery of itemsets as its level of complexity is significantly greater than that of inference generation. Given E distinct items within the search space, there are $2^{|E|}$ possible combinations of items to explore and given that $|E|$ is often large, naive exploration techniques are often intractable.

Relevant research can be organized into four groups:

—constraining the exploration, through the development and incorporation of *measures of interest* (MOI) and efficient pruning strategies;

—reducing I/O, through hardware advances, enabling large datasets to become memory resident, and techniques such as intelligent sampling;

—creating useful data structures to make analysis more tractable. (This research, initially driven by the need to reduce I/O, has resulted in an evolution of structures to efficiently represent the exploration space.);

—producting condensed inference sets allowing the entire result set to be inferred from a reduced set of inferences, reducing storage and facilitating user interpretation.

Over the past decade a variety of algorithms have been developed that address these issues through the refinement of search strategies, pruning techniques, data structures, and the use of alternative dataset organizations. While most algorithms focus on the explicit discovery of all inferences for a given dataset, increasing consideration is being given to specialized algorithms that attempt to improve processing time or facilitate user interpretation by reducing result set size and incorporating domain knowledge.

This article, therefore, presents a survey of association mining fundamentals, detailing the evolution of generic association mining analysis algorithms from the seminal Apriori algorithm [Agrawal et al. 1993] to the state-of-the-art. Rather than include an analysis of the different measures of interest that can be applied to algorithms or the more recent areas of algorithmic specialization, the article provides a substantive summary of the foundation algorithms on which they are based. The survey also addresses the important issue of generic optimizations, namely, condensed representations and incomplete set algorithms which we consider as optimizations applicable to fundamental itemset, and inference discovery algorithms which reduce analysis time and produce reduced sets for subsequent inference derivation or reporting.

This survey is organized in the following way. As the majority of research to date has focused on the identification of itemsets that are valid in respect to specified constraints,

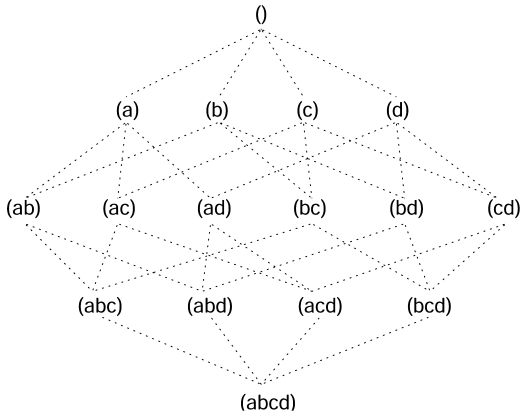


Fig. 1. Search space lattice.

it forms the bulk of the review and has been separated into three sections. The first, itemset identification discussed in Section 2, provides the foundation for the review introducing general concepts, presenting common notation, and discussing dataset organization. This is followed by a comprehensive survey of classic algorithms in Section 3 and a survey of fundamental optimizations, namely, condensed representation algorithms in Section 4 and incomplete set algorithms in Section 5. Each section is accompanied by a table summarizing the algorithms. Section 6 completes the survey with a discussion on the derivation of inferences from the identified valid itemsets, including inferencing techniques.

2. ITEMSET IDENTIFICATION

The identification of valid itemsets is computationally expensive, requiring the consideration of all combinations of distinct items, E, (or $2^{|E|}$) subsets, resulting in exponential search space growth as $|E|$ increases. This is illustrated in Figure 1 which shows the search space lattice resulting from $E = \{a, b, c, d\}$. Furthermore, it is the consideration of this lattice in the presence of a specific dataset D that results in an often intractable analysis scenario due to the size of the required structures and the fine grained exploration required. Itemset identification research thus focuses on reducing dataset I/O and on constraining the exploration. There are four applicable classes of I/O reduction suggested in the literature: projection, partitioning, pruning, and access reduction.

—*Projection*. The projection of D onto an equivalent condensed representation reduces storage requirements, possibly enabling memory residency, and may also result in computational optimization through algorithmic exploitation of this new representation. For example, the projection of a corpus into integer or binary representation may result in improved processing time as numeric comparison is computationally faster than string comparison.

—*Partitioning*. Dataset partitioning minimizes I/O costs by enabling the memory resident processing of large datasets, reducing costly disk access.

—*Pruning*. Dataset pruning techniques dynamically reduce the dataset during processing, discarding unnecessary items or objects. This results in a reduction of D, possibly to the point at which D can achieve memory residency, further reducing processing time.

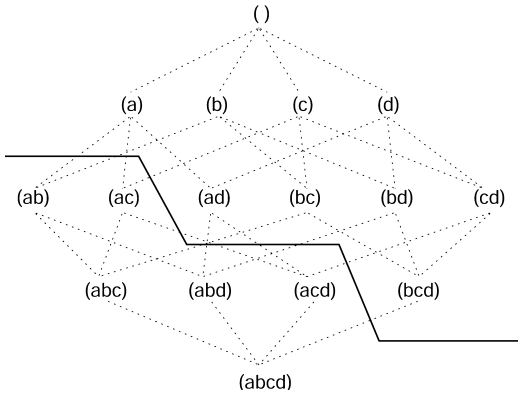


Fig. 2. Bounded search space lattice.

—*Access Reduction*. Reduction in the number of times that disk resident datasets need to be accessed to identify all itemsets, reducing processing time.

Constraint of the search space through user-specified heuristics and domain constraints can significantly reduce exploration while improving the quality or interest of the resultant inferences. The most common constraint used to reduce exploration during itemset identification is the specification of a support threshold *minsup*, where support (σ) relates to the number of times that an itemset appears within D. Therefore, an itemset e, such that $e \subset E$, if $e < minsup$, is considered not of interest and is removed from subsequent consideration. This effectively reduces itemset identification to the discovery of only those itemsets that exceed a specified presence within D, or the discovery of *valid* itemsets.

If a constraint is reflexive, or directed, its inclusion can provide significant optimization due to the definition of a boundary within the search space lattice, above or below which exploration is not required (shown in Figure 2). In relation to itemset identification, a reflexive constraint is one that never decreases (*monotonic*) or increases (*nonmonotonic*) as the number of items within an itemset increases. Reflexive constraint inclusion can, therefore, result in rapid search space reduction by effectively eliminating all supersets or subsets of an invalid itemset. For example, the heuristic constraint *support* is nonmonotonic as given an invalid itemset, $\sigma(e) < minsup$, all supersets of e can be eliminated from consideration. With respect to support in particular, this reflexive effect was first introduced as the *Apriori* heuristic due to its initial inclusion within the Apriori analysis algorithm (see Section 3) and is also commonly known as the Downward Closure Principle (DCP). This review typifies constraint inclusion through the use of *support* as it is the concept of constraint inclusion rather than its actual nature that is of interest . A more formal description of these common measures of interest used in association mining analysis are presented in the following section.

2.1. Notation

Let $U = \langle O, E, D \rangle$ be the universal association mining context, where O and E are finite sets of objects and items, respectively, and E is distinct. $D \subseteq O \times E$, is a binary relation, commonly referred to as the dataset, such that the existence of the couple $< o, e > | o \in O \wedge e \in E$ denotes that e is related to o. Given the sets $x, y \subseteq E$, commonly termed itemsets, such that $x, y \subseteq o$, then $t(x)$ be is the subset of O containing x, and $k = |x|$. An inference r is of the form $x \Rightarrow y | x, y \subset E \wedge x \cap y = \emptyset \wedge x, y \neq \emptyset$.

Given that the algorithm incorporates the fundamental quality heuristics of support and confidence, then support $\sigma(x)$ is defined as the fraction of objects O that contain itemset x with respect to D ($\sigma(x) = |t(x)|/|D|$), while confidence γ, or inference strength, is the frequency of O containing x that also contains y ($\gamma(x \Rightarrow y) = \sigma(x \cup y)/\sigma(x)$). Alternatively, confidence can be represented as $\gamma(x \rightarrow y) = P(y|x)$, where P is the conditional probability. Thus the support of an inference is the fraction of O that contains the x union y, $\sigma(x \Rightarrow y) = |t(x \cup y)|/|D|$.

Taking these quality heuristics into account results in constraining itemset identification to the discovery of all valid itemsets V, such that $V = \bigcup_i \{x | x \subseteq E | \sigma(x) \geq minsup\}$, where *minsup* is the specified minimum support threshold above which an itemset is considered frequent. The inferences are then derived from the permutations of each V_i such that the inference $x \rightarrow y$ satisfies the conditions $x \cup y = V_i, x \cap y = \emptyset, \emptyset \neq x \neq y, \gamma(V_i) \geq minconf$.

2.2. Dataset Organization

Association mining data is generally obtained from databases created for other uses and massaged into a suitable representation through preprocessing techniques. The resulting dataset is expressed as either a sparse vector matrix or a set of enumerations and can be organized in regard to the objects or items that they contain.

Within the vector matrix, each cell represents the existence or absence of an item (binary) within a particular object, while an enumerated format is a condensed representation containing only items positively associated with an object. The preferred format to use is largely dependant on dataset density since although the vector format is computationally faster, as combinational processing is reduced to logic operators, the enumerated format becomes more viable as the dataset density decreases. For example, given a dataset in which $|E|$ is large and each object only contains a few items, the enumerated format will require significantly less space and fewer combinatoric operations during itemset identification. Additionally, the enumerated format can be constrained to a binary representation consisting of ordered enumerated pairings of object and item identifiers, hence if horizontally organized, an object will be represented over $|\{e|e \in o\}|$ rows.

Dataset organization is typically horizontal as each row contains an object, while vertical organization refers to the representation of objects as columns and with each row within the dataset representing an item. Since a dataset is normally processed on a row, by, row basis, horizontal organization is said to maintain an *object focus*, while vertical organization maintains an *item focus*. Both organizations can be represented using either an enumerated or vector format where they are referred to as *Tidlists* and *Tidsets*, respectively, when using vertical organization where the Transaction Identifier (Tid) refers to the contents of each row. Previous research has shown that the vertical organization can lead to more efficient algorithms as the search for item-based inferences is better served by an item-focused layout. This was summarized by Shenoy et al. [2000] who describe four advantages of the vertical organization over the horizontal organization.

—*Improved itemset validation*. Itemset validation is computationally faster in a vertical layout due to its item focus. For example, using *support*, the validation of ab using vertical organization is achieved by finding the size of the intersection of two items a and b, Equation (1). Using a horizontal organisation, validation becomes the sum of those objects that contain both a and b, requiring a full scan as the items are scattered throughout D, Equation (2).

$$\sigma(ab) = |a \cap b| \tag{1}$$

$$\sigma(ab) = |\{d | a \in d \wedge b \in d \wedge \forall d \in D\}| \tag{2}$$

—*Automatic dataset reduction*. Given the inclusion of nonmonotonic constraints, items can be easily removed from the dataset, reducing its size.

—*Improved vector compression.* Vector compression is greater in larger datasets and hence better in a vertical layout as typically the number of objects exceeds the number of items.

—*Asynchronous computation.* Given nonmonotonic constraint inclusion, vertical organization allows the computation of an itemset once its subsets have been deemed valid, whereas using horizontal layout all itemsets must be computed before any supersets are considered.

Recent attention has been given to the optimization of vertical vector representation for sparse matrices through projection, reducing storage and processing. Burdick et al. [2001] propose a technique whereby vector reduction is achieved by projecting onto a smaller vector, consisting of positive associations only, when an itemset's support reaches a defined rebuilding threshold. The new vector is then used to calculate the support for the itemset's supersets by applying the new projection, optimizing support calculation through vector reduction. Shenoy et al. [2000] introduce compressed vertical bitmaps or *snakes* that reduce the vertical representation in comparison to the equivalent horizontal representation. Additionally, their proposed algorithm, VIPER, uses a combination of horizontal and vertical layouts during processing. Zaki and Gouda [2001] introduce the concept of *diffsets* that only keep track of the differences in the Tidlists of an itemset from its generating valid itemsets. This technique is particularly efficient when dealing with dense datasets as the difference between the Tidlist of an itemset and that of its direct superset is often significantly smaller than the Tidlist itself.

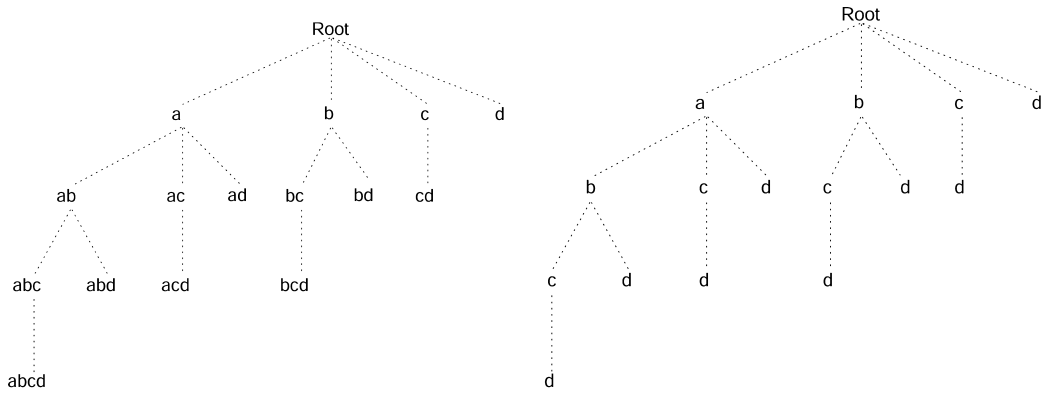
Dataset organization also influences the type of lattice traversal strategy used to explore the search space. Given the two options of either breadth-or depth-first, BFT and DFT, respectively, BFT traversals can use either organization format. However, DFT is optimized for the use of a vertical organization. BFT strategies generate all valid itemsets of size κ, V^{κ} before generating V^{κ}. Only a single scan of D is therefore required for each κ using BFT and both organization formats are acceptable. However, using DFT and a horizontal organization, a scan of D is required for each recursive call during the traversal to validate the candidate itemsets, resulting in large processing overheads to validate a small number of itemsets, whereas using a vertical organization, a full scan of D is not required.

3. CLASSIC ALGORITHMS

This section discusses the classic itemset identification algorithms, which are regarded as those that discover all distinct valid itemsets within a dataset. Participant algorithms can be separated into two classes, candidate generation and pattern growth, within which further division is based upon traversal and underlying data structures. The majority of classic algorithms are candidate generation, where candidate itemsets are constructed and then validated. Pattern growth techniques however, eliminate the need for candidate generation by constructing complex hyperstructures that contain representations of the itemsets within the dataset.

3.1. Candidate Generation Algorithms

Candidate generation algorithms identify candidate itemsets before validating them with respect to incorporated constraints, where the generation of candidates is based upon previously identified valid itemsets. The core algorithm of this genre is Apriori, on which many later algorithms are based.



(a) Enumeration Set (b) Prefix Tree (trie)

Fig. 3. Common storage structures.

This section provides a survey of candidate generation algorithms, providing a detailed description of the core algorithms and highlighting the significant contributions of others. First, there is a short discussion on the common data structures used within candidate generation algorithms. The article then splits into a discussion of merge- and extension-based algorithms that roughly correlate to BFT and DFT. Finally, there is a section relating to hybrid algorithms that attempt to provide further optimizations by switching between traversal and accrual methods during processing.

3.1.1. Tree Data Structures. Candidate generation algorithms generally use tree-based data structures of which there are three common types—hash trees, enumeration-set trees, and prefix trees.

Hash-trees, a combination of b-tree and hashtable structures, was introduced by Agrawal and Srikant [1994] as an effective candidate itemset storage structure. The structure is effectively a b-tree for which every internal node is a hashtable, and every leaf node or bucket contains a set of itemsets. When a bucket reaches its quota of itemsets, the hash-tree extends by replacing the bucket with a new hashtable into whose buckets the itemsets are placed.

The enumeration-set tree [Rymon 1992] is an ordered tree where each node n represents an itemset, $e \in V$, and an edge represents a single item extension of that itemset. Each tree level therefore contains itemsets of the same length. For example, third-level nodes contain valid itemsets of length three (V_3).

An enumeration-set tree evolves through the single item extension of leaf nodes, each resulting in a new child node. This process is ordered and constrained so that, given a particular item ordering, only those items occurring after the last item of the current item set are considered during extension of the current itemset. Given this subset of possible extensions, the process is further constrained so that only new valid itemsets are inserted into the tree. The presence of reflexive constraints, in this case nonmonotonic constraints, further reduces the set of possible item extensions to those items that resulted in valid extensions of its direct ancestor (parent node).

For example, given a node n and an ordered item set E, then n's extension set, n^{ext} consists of all $e \in E$ that occur after e_i where $n = \{e_1, e_2 \ldots e_i\}$. This process is illustrated in Figure 3(a) for the item set $E = \{a, b, c, d\}$. Furthermore, if the process is nonmonotonically constrained, then n^{ext} is further constrained to those valid extensions of its parent p.

Prefix trees, also known as tries, (Figure 3(b)), although fundamentally equivalent, differ from the enumeration-set tree in that each node specifies the projected prefix

Algorithm 3.1 Apriori Itemset Generation

Apriori itemset Generation

1: $\kappa = 0$
2: **repeat**
3: κ^{++}
4: **if** $\kappa > 1$ **then**
5: $\{C_\kappa = \text{generate\_candidates}(V_{\kappa-1})\}$
6: **else**
7: $\{C_1 = E\}$
8: **endif**
9: **for all** O **do**
10: $C_o = \{c \mid c \subseteq o \wedge c \in C_\kappa \wedge o \in O\}$
11: **for all** C_o **do**
12: $C_\kappa^i.count^{++} \mid C_o \in C_\kappa$
13: **end for**
14: **end for**
15: $L_\kappa = \{C_\kappa \mid C_\kappa^i.count \geq minsup\}$
16: **until** $V_\kappa = \emptyset$
17: return $L = \bigcup_{k=o}^{n} L_\kappa$

item for a particular subtree instead of the entire prefix. This is evident in Figure 3 where the two trees are structurally equivalent, the difference is that, while within the enumeration-set tree all itemset information is contained at the node, within the prefix tree, the same information is accrued during traversal to the node (itemset) in question.

3.1.2. Merge Algorithms. The fundamental merge algorithm is Apriori [Agrawal and Srikant 1994], commonly regarded as the classic association mining algorithm, which introduced reflexive constraint inclusion, namely *support*, to reduce exploration. The algorithm derives candidate itemsets C_κ from $V_{\kappa-1} \mid \kappa > 1$, incorporating *support* to reduce $|C_\kappa|$. From this, V_κ is derived through a scan of the dataset, accruing counts for each $c \in C_\kappa$. Thus given a *support* threshold $minsup$, $V_\kappa = \{C_\kappa : \sigma(C_\kappa) \geq minsup\}$.

Algorithm 3.1 presents the iterative Apriori algorithm that requires κ dataset traversals. The algorithm has two main parts, candidate generation and validation (accrual). The set of candidates is formed by the set of items, E, given $\kappa = 1$, otherwise it is based on a merge function involving members of $V_{\kappa-1}$. Subsequent accrual determines the candidate itemset *support*, and those meeting the nominated threshold $minsup$ are appended to the valid set V_κ. Thus $V_1 = \{e \mid e \in E \wedge \sigma(e) \geq minsup\}$.

Subsequent $L_\kappa \mid \kappa > 1$ is based on $V_{\kappa-1}$, given the inclusion of *support*, so that an itemset can only be valid if all subsets are valid. For example, given a valid itemset $a \mid a \in V$, then all subsets of a also exist in V. Given this, C_κ is constructed from the constrained merge of $V_{\kappa-1}$ pairs where the pairs only differ by a single item, and all other $\kappa - 1$ permutations of the new itemset also exist in $V_{\kappa-1}$.

For example, given two itemsets $a = \{a_1, \ldots a_n\}$ and $b = \{b_1, \ldots b_n\}$, a merge only occurs if a and b's last items differ (Equation (3)). The newly created candidate itemset $c \mid c = \{a \cup b\}, |c| = k$ is then appended to the set of candidate items C_κ if all subsets of length $k - 1$ exist in $V_{\kappa-1}$. Given that $V_3 = \{abc, bdg, abf\}$, since abc and abf differ by only their last item, a new itemset $abcf$ is constructed. However, since not all V_3 subsets of $abcf$ in $V_{\kappa-1}$ exist (e.g. bcf), $abcf$ is not appended to C_κ as it cannot be valid due to the reflexive *support* constraint.

Once the set of candidates, C_κ, has been constructed, accrual is undertaken to find the valid candidates to be appended to V_κ, using the same accrual technique as for V_1. This process repeats, with itemset length incrementing, until the newly generated V_κ is empty.

$$c = \{a \cup b | a, b \in V_{\kappa-1} \wedge \{a_1, \ldots a_{n-1}\} = \{b_1, \ldots b_{n-1}\} \wedge a_n \neq b_n\} \tag{3}$$

Independent association mining research occurring at the same time [Mannila et al. 1994] also came to the same conclusion as Agrawal and Srikant in regard to the reflexive constraint inclusion. However, this implementation, Offline Candidate Determination (OCD), is less efficient as the join conditions are not as tightly constrained, resulting in superfluous sets. Two join operations were devised, the first variant (Equation (4)) specifies that any two members could differ, while the second variant (Equation (5)) extended each member of $V_{\kappa-1}$ with every member in V_1.

$$c = \{a \cup b | a, b \in V_{\kappa-1} \wedge |a \cap b| = k - 2\}, \tag{4}$$

$$c = \{a \cup b | a \in V_{\kappa-1} \wedge b \in V_1 \wedge b \not\subseteq a\}. \tag{5}$$

The realization of reflexive constraints enabled Apriori to surpass earlier frequent itemset algorithms, namely, AIS and SETM. AIS [Agrawal et al. 1993] creates V_κ from $V_{\kappa-1}$ by scanning D and, for each itemset $a \mid a \subset o \wedge o \in O \wedge a \in V_{\kappa-1}$, a new group of candidate itemsets are considered by extending a with each item in o that lexicographically occurs after the last item of a. If the new itemset already exists, then its count is incremented; otherwise it is appended to C_κ with a count of 1. After parsing O, $V_\kappa \subset C_\kappa \mid C_\kappa^i.count \geq minsup$.

SETM [Houtsma and Swami 1993] also generates candidate itemsets on the fly. However the need to iterate through an object's $V_{\kappa-1}$ permutations to find itemsets is alleviated by separating the algorithm's candidate generation and counting functions. During candidate generation, the algorithm stores each candidate itemset and its object identifier in an array-based structure A_κ. Once the dataset scan is complete, candidate validity is determined by sorting and aggregating A_κ, removing any A_κ^i, that do not meet the criteria. The subsequent generation of $A_{\kappa+1}$ is facilitated by A_κ as all valid κ permutations of $o \in O$ are readily available.

The advent of Apriori and reflexive constraint inclusion provided the standard pruning or search space reduction technique, mainly through the use of the quality heuristic *support*. Subsequent research in analysis optimization focused on reducing I/O through condensed representations, dataset partitioning, dataset pruning,and dataset access reduction.

AprioriTID [Agrawal and Srikant 1994] extends Apriori by eliminating multiple scans of D through the construction of a counting-base set $\bar{C}_\kappa$, during construction of V_1. The counting base is of the form $< Tid, C_\kappa^i >$, where Tid is the object identifier and C_κ^i denotes the set of C_κ present in an object $o \in O$. For example, if $o = \{a, b, d, f\}$, then equivalently $\bar{C}_1^i = \{\{a\}, \{b\}, \{d\}, \{f\}\}$; however, $\bar{C}_2^i$ contains all potential V_2 within the object o, hence $\bar{C}_2^i = \{\{ab\}, \{ad\}, \{bd\}, \{bf\}, \{df\}\}$. It is apparent from this example that $\bar{C}_\kappa$ may potentially be larger than D for small $\kappa > 1$, however, this is quickly reduced as κ increases as if o does not contain a C_κ, then $\bar{C}_\kappa$ will not have an entry for that object, in effect incorporating a form of dataset pruning.

Another Apriori extension proposed by Agrawal and Srikant [1994] is delayed accrual (or pass bundling [Mueller 1995]) which is based on the observation that any $\sigma(C_i) \mid |C_i| = \{\kappa + 1, \ldots 2\kappa\}$ can be represented as the union of some pair of V_κ itemsets.

Thus from a single scan of D, the *support* of all candidates of length $\kappa + 1...2\kappa$ can be computed. A trade-off exists, however, between the time saved by reducing the dataset access and the number of false positives generated through the projection of C_κ instead of V_κ.

Dynamic Itemset Counting (DIC) [Brin et al. 1997] reduce I/O by enabling the calculation of V_κ itemsets during earlier scans, $\leq \kappa$, through dataset partitioning and the use of checkpoints. If during processing all immediate $\kappa - 1$ subsets of a larger itemset $a \in C_\kappa^i$ have been determined valid at a checkpoint (end of a dataset partition), then the calculation of $\sigma(a)$ can begin and will terminate when it reaches the same checkpoint during the next dataset iteration. This reduces dataset access as, within a single scan, itemsets of differing lengths can be counted where the length of the active itemsets is always $\geq$ the current scan number.

—*Dataset-global pruning* is based on the premise that, if an item participates in V_κ^i, then it must participate in $|\kappa - 1|\ V_{\kappa-1}$ itemsets. Thus any e that does not appear in at least $|\kappa - 1|\ V_{\kappa-1}$ itemsets can be removed from o. If this process subsequently results in $|o| < |\kappa|$, then o is removed from D.

—*Dataset-local pruning*, also used in DHP, is applied to each object during subset counting. It states that an item e can only participate in an itemset $V_{\kappa+1}^i$ if e exists in $|\kappa|V_\kappa$ subsets within o. However, since V_κ is unknown and since $C_\kappa \supseteq V_\kappa$, then C_κ is used instead. Hence any item $e \mid |e \in c| < \kappa \wedge c = |C_\kappa \in o|$ is removed from the object.

Direct Hashing and Pruning (DHP) [Park et al. 1997], introduces the concept of *possible frequent itemsets* C'_κ to optimize the generation of candidate itemsets and to provide a novel dataset trimming technique. The possible frequent itemset $C'_{\kappa+1}$ accrues the count for each possible $C_{\kappa+1}$ itemset within each $o \in O$ for which all κ subsets exist in C'_κ during the construction of V_κ. This in effect accumulates information about C_{k+1} in advance so that all possible C_{k+1} itemsets are encoded within the C'_{k+1} hashtable and results in the production of a significantly smaller but accurate candidate set C_κ. However, because of the possible collisions on C'_{k+1} entries, each derived C_κ^i still needs to be checked for *minsup*. The algorithm also incorporates progressive dataset pruning to discard items and objects that are of no further use. The two pruning techniques, dataset-global and dataset-local (presented previously), are incorporated during accrual to reduce $|D|$ during each scan. DHP also incorporates the concept of delayed accrual, discussed earlier. The incorporation of progressive trimming, delayed accrual, and the identification of C'_κ results in significant speedup over Apriori for small κ, due in the main to the reduced size of C_κ.

Perfect Hashing and Pruning (PHP) [Ozel and Guvenir 2001] proposes an optimization to DHP by using perfect hashing in the creation of the hashtable for $C'_{\kappa+1}$. This effectively eliminates the hashtable collisions that were apparent in DHP and hence $C'_{\kappa+1}$ will contain the actual counts of the $C_{\kappa+1}$ itemsets alleviating the need to recount the occurrence of $C_{\kappa+1}$ itemsets in D.

Direct Count and Prune (DCP) [Orlando et al. 2001b] attempts to optimize itemset discovery by incorporating the dataset pruning techniques introduced in DHP and by using direct counting in the validation of candidates. Instead of the usual hash-tree structure to perform accrual, DCP uses a *direct counting* method. For small κ, the hash-tree structure is ineffective as C_κ is generally large and, because tree depth is based on κ, only a few partitions will be created. The direct count method is a generalization of Apriori's accrual technique, differing for $\kappa \geq 2$.

V_2 generation, based on V_1, uses a vector *Counts* to accrue each C_2^i candidate pair, in lexicographical order, using a perfect hash function as shown in Figure 4(a). The counting of subsequent C_κ is accomplished by extending *Counts* to create a directly

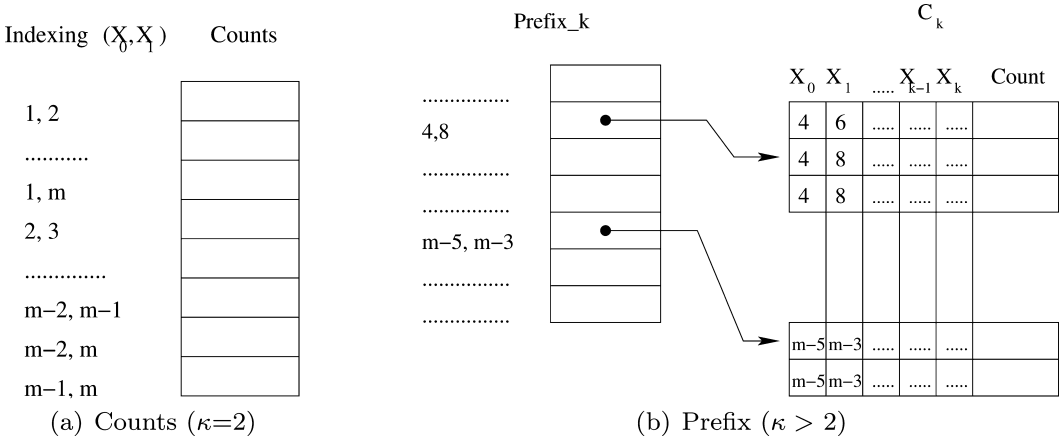


Fig. 4. DCP: data structures.

accessible prefix tree of shared 2-prefix. This allows direct access to the subset of candidates specified by their 2-prefix. This structure exploits spatial locality in that, once a prefix $\{a_{i_1}, a_{i_2}\} \mid a_{i_1} < a_{i_2}$ is selected, the possible κ completions to the prefix existing in o will be found in the subsequent contiguous section of C_κ.

The Partition algorithm [Savasere et al. 1995] discovers all valid itemsets in two dataset scans. This is accomplished by partitioning the dataset, each of which fits into memory and then finding the valid itemsets that exist in each partition p. The theory that any globally valid itemset V_e must be valid within at least one partition V^p. During the first dataset scan, all valid itemsets for each partition V^p are generated using AprioriTid, the vertical organization of which is constructed during the generation of V_1^p. This is achievable as each partition can fit in memory, and hence D is only scanned once during the generation of V_p. Based upon this, the global set of candidates C is derived from the union of V_κ^p (Equation (6)). The second dataset parse determines V from C through accrual of each C_i within each partition.

$$C = \bigcup_{\kappa=1}^{n} c_\kappa \mid C_\kappa = \bigcup_{p=1}^{n} V_\kappa^p \tag{6}$$

SPINC and AS-CPA algorithms provide optimizations to the Partition algorithm. SPINC [Mueller 1995] reduces process time by incrementally constructing C, adding V_κ^p to C whenever it is available instead of waiting until the end of the first scan as in Partition. This allows SPINC to start global occurrence accrual for each C_i during the first scan, potentially resulting in scan reduction. AS-CPA [Lin and Dunham 1998] attempts to eliminate any data skew apparent in Partition's results. Data skew refers to the irregularity of data distribution over D that may cause the generation of false candidates. For example, seasonal trends, in which a product's sales are high for a short period of time only, may only meet a support threshold for a particular temporal partition but not globally. AS-CPA achieves skew reduction by introducing early local pruning and through two global antiskew techniques. Early local pruning is based on *better* local valid itemsets B_κ^p, where B_1^p is derived from $V_1^p \cap V_1^{2\ldots p}$ and used to construct the local set of valid itemsets (B_κ^p) using regular Apriori. Since $B_1^p \subseteq V_1^p$, generally $|B_\kappa^p| < |V_\kappa^p|$, resulting in reduced processing. The two global antiskew techniques refer to the first and second dataset scans and assume the incorporation of SPINC's incremental C construction within the algorithm. They are based on the premise that,

if an itemset is not valid over a number of partitions, then it should be removed from consideration. If during further processing, the same itemset is found to be valid within a partition, it can again be considered potentially valid (see Mueller [1995] for a full discussion).

Hierarchical Bit Maps (HBM) [Gardarin et al. 1998] uses a vertically-organized vector format, calculating support efficiently by performing 2-way intersections of relevant bitmaps. The algorithm encodes the bitmap into an unsigned *short* by creating groups of 16 bits upon which the subsequent intersection is performed. It was realized, however, that, in a typical mining run. many *short*s were empty due to sparsity, hence performing the intersection calculation was inefficient. To alleviate this, a second level of bitmap indexing was created, in which each bit represents whether or not a *short* is empty. Hence, when calculating the support of two itemsets, an intersection of the second-level indexing is done first, providing a list of the nonzero *short*s to consider further. Prior to HBM, Gardarin et al. [1998] developed NBM which used the same format but did not make use of the second level of indexing.

ColumnWise [Dunkel and Soparkar 1999] uses a column-bsed instead of row-based iteration of D, hence, although d remains horizontally-organized, it uses intersections in the construction of C_κ. The actual processing is similar to Apriori using a horizontal bit vector layout and intersection accrual. If the results of prior $(\kappa - n : n > 1)$ joining operations are preserved, $\sigma(C_\kappa^i)$ is the intersection of 2 immediate subsets of C_κ^i, otherwise a κ-way join of each participant item is required.

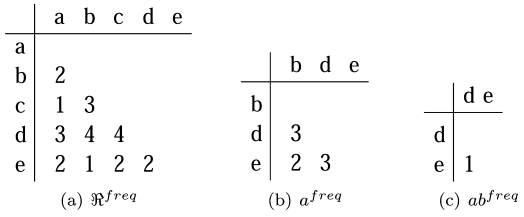
DLG [Yen and Chen 1996] is a novel graph-based technique based on a lexicographically-ordered vertical bit vector layout. Through bit vector summation, all infrequent items are removed resulting in V_1 from which the association graph is constructed. The algorithm constructs an edge between any two items, a and $b : a < b, |a \cap b| \geq minsup$, effectively resulting in the identification of V_2. Subsequent $V_\kappa, \kappa > 2$ generation is based on direct extension where the edges from the last node of V_κ^i, are used to create $C_{\kappa+1}$. If no edges exist, then no $\kappa + 1$ itemsets based upon it can exist. For example, given a valid V_κ itemset $a = \{a_1, a_2 \ldots a_n\}$, if no additional directed edges from a_n exist apart from a_{n-1}, then no $V_{\kappa+1}$ that are supersets of a can exist. Thus for all directed edges from a_n, a $C_{\kappa+1}$ is generated, the actual support for which is calculated through the κ-way intersection of participants.

3.1.3. Extension Algorithms. Extension algorithms rely upon the single item extension of valid itemsets to derive new candidate itemsets. This section discusses the contributions of various extension-based algorithms, providing a discussion of core algorithms and summarizing the contribution of other algorithms that extend these core algorithms.

Tree Projection [Agrawal et al. 1999] discovers valid itemsets through the construction of an enumeration-set tree and by incorporating object projection and matrix-based accrual techniques. The enumeration-set tree is built in the typical DFT manner described in Section 3.1.1. Matrix-based accrual is facilitated through object projection where only objects relevant to a node's itemset are projected to the node.

Tree Projection is initiated by the regular BFT generation of V_1 which forms the basis of the enumeration-set tree and the initial extension set of the root node $\Re^{ext}$. From this, the frequency matrix $\Re^{freq}$ is constructed and populated by $\Re^{project}$, which is a projection of D containing only those items in $\Re^{ext}$. Object projection is progressive so that n^{freq} is based on $n^{project}$, which has been derived from $\bar{n}^{project}$, and filtered to only those items in n^{ext}. The technique of projecting the relevant object information at each node significantly reduces processing through object reduction.

Subsequently the calculation of a node's extension set n^{ext} is derived from its parent's frequency matrix $\bar{n}^{freq}$, within which each cell contains the co-occurrence frequency



(a) $\Re^{freq}$ (b) a^{freq} (c) ab^{freq}

Fig. 5. Tree projection: matrices.

of the intersecting items within $\bar{n}^{project}$. Given $minsup = 2$, Figure 5 presents a set of related matrices for $\{r, a, ab\}$ from which it can be seen that the extension set of a ($a^{ext} = \{b, d, e\}$) is derived from rows in column a of $\Re^{freq}$ that are greater than or equal to *minsup* and which lexicographically occur after item a. From this, the projected dataset $a^{project}$ is derived and subsequently a^{freq} constructed. This recursive process continues until $x^{ext} = \emptyset$. The resulting constructed enumeration-set tree uniquely identifies all valid itemsets where validity (based on frequency) is derived from projected matrices.

The authors additionally propose different tree construction techniques based on BFT and DFT. Using BFT, all $n^{ext}_{stikappa}$ are found before $n^{ext}_{\kappa+1}$ by reinitiating the object projection process for each subsequent level of the tree. Each object $\Re^{project}$ is, in turn, recursively projected to all $n_{\kappa-1}$ matrices. Using DFT, the projected datasets are maintained for all nodes and their siblings on the path from the root node to the node currently being extended. Hence matrix creation becomes an independent problem with a significantly reduced dataset. The choice of strategy relates to a trade-off between I/O and processing; BFT requires more processing as each object must be reprojected for each successive tree level, while DFT requires that all projected datasets be maintained. While BFT is more efficient in the discovery of short itemsets, DFT is better for finding long itemsets. A hybrid technique is also suggested that uses BFT until each projected dataset for a tree level can be held in memory at which point each projected dataset is saved to a separate file, sequentially read into memory, and independently processed using DFT, discovering all descendants.

In the pursuit of optimizing long itemset discovery, the same authors developed Depth Project [Agrawal et al. 2000]. The algorithm has similar foundations in its reliance upon the enumeration-set tree and a DFT strategy, but differs in regard to dataset projection occurrence and accrual. The algorithm uses a bit-vector structure in which each bit represents a projected object to provide efficient accrual—if a node's value or itemset is a subset of an object the relevant bit is turned on. A node's bit-vector is derived from its parent's bit vector by turning off all objects that do not contain the extension item. This structure is ineffective where $|E|$ is much larger than the average number of items within an object because most bit's will be off. However, it becomes efficient due to the reprojection of the objects when a minimum on threshold is reached that results in the derivation of a new, reduced vector with all item bits on. Furthermore, Depth Project proposes the generation of the extension set n^{ext} that is similarly derived from $\bar{n}^{ext}$ without using projected matrices. The projected dataset $n^{project}$ is first filtered so only valid $\bar{n}^{ext}$'s objects remain. If there exists a large number of distinct projected objects at n, accrual is through byte counting of $a \mid a \in \bar{n}^{project}$. If the distinct projected set is small, typically at lower tree levels, the repetitiveness is exploited through a novel bucketing technique that counts the entire subtree in a single projected dataset scan.

Pijls and Bioch [1999] propose apriori-*DF* a novel trie extension technique in which the valid subtree of V_1^i is used to calculate the valid subtree of V_1^{i-1}. The algorithm constructs V_1 in regular Apriori fashion and stores them in order of ascending frequency.

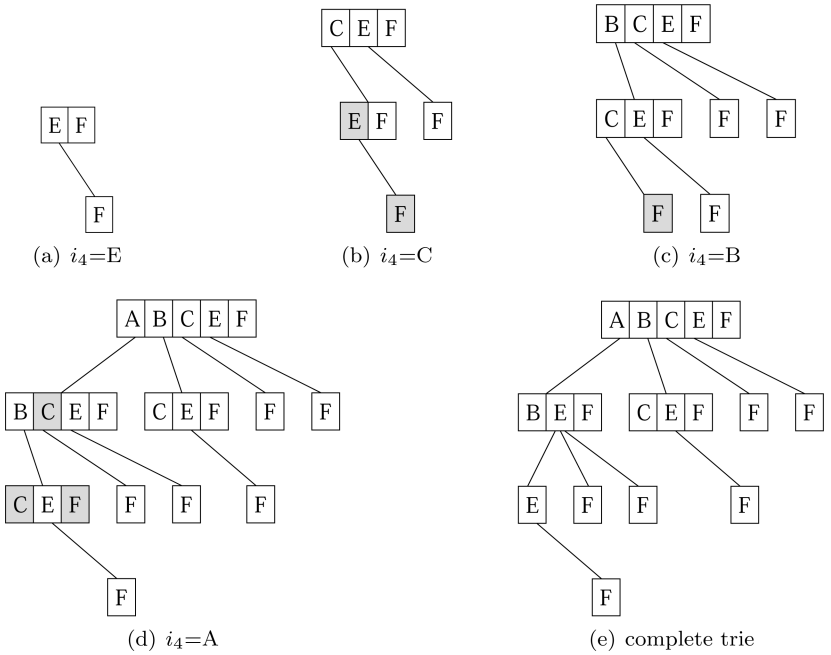


Fig. 6. Apriori-DF trie construction with invalid candidates highlighted.

Subsequent processing is undertaken from most to least frequent (left to right) with a dataset scan required for each V_1 member, therefore, a total of $V_1 - 1$ scans of the dataset are required. This process, given the ordered set V_1 = {A,B,C,E,F}, is highlighted in Figure 6 (from Kosters and Pijls [2003]), showing the replication of V_1^n's subtree under V_1^{n-1} which provides the candidate set based on V_1^{n-1}. Any invalid nodes are then removed from this subtree, by performing dataset accrual. A subsequent implementation described by Kosters and Pijls [2003] claims to further optimize the algorithm through memory management by undertaking accrual in the subtree before appending it to V_1^{n-1}, thereby alleviating the need to delete invalid itemsets.

Eclat and Clique [Zaki 2000] are similar techniques that improve valid itemset discovery by recursively decomposing an association lattice into disjoint subsets until each can be independently solved in memory, reducing I/O. Both algorithms use the association lattice, BFT traversal, and the same accrual technique, but differ in regard to the method of decomposition.

The association lattice used by these algorithms is derived from V_2, which is constructed using a classical Apriori approach. Within the association lattice, each item is represented by a node, and edges between nodes represent the presence of an association within V_2 (Figure 8(a)). Typical organization is vertical and hence accrual is based on Tidlist intersections for each sublattice constructed. The authors also introduce a variation (AprClique) that uses a horizontal dataset organization, and hence uses counting instead of accrual. Furthermore, Eclat's accrual is undertaken in reverse lexicographical order to optimize pruning.

Eclat decomposes the search space lattice using the concept of prefix-based equivalence relations, where members of a relation share the same κ-prefix. For example, given $V_2 = \{ab, ad, bc, bd, cd\}$, the resulting equivalence classes are $[a] = \{b, d\}, [b] = \{c, d\}, [c] = \{d\}$. Furthermore, each equivalence class is a lattice in its own right and can undergo recursive decomposition until each sublattice fits into memory. Figure 7

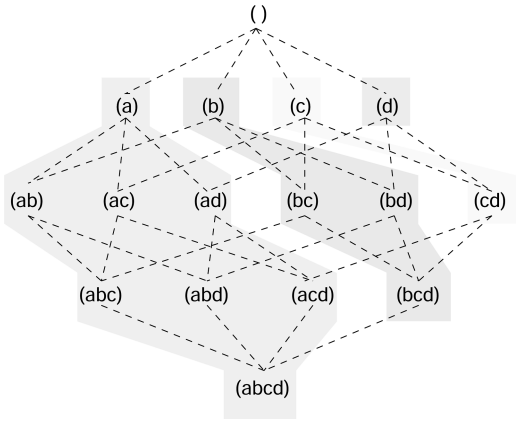


Fig. 7. Eclat: equivalence classes.

illustrates the search space lattice presented in Figure 1 segregated into equivalence classes.

Once decomposed, all valid itemsets are then discovered in each class using a BFT Tidlist intersection strategy incorporating *support*. In addition, it is apparent from Figure 7 that dependencies exist between the classes, that is acd is dependant on cd as well as subsets from within its own class. By solving the classes in the reverse order to which they were created, all subset information required for effective pruning is available.

Clique decomposes the search space using the concept of maximal cliques. A clique is a complete lattice where completeness is denoted by an edge between all pairs of vertices. A maximal clique set is then the set of cliques that represent the search space lattice within which no clique is a superset of any other. This technique uses additional information to form smaller sublattices than Eclat's prefix-based approach and hence is often more effective as less accrual is required.

Maximal clique enumeration is based on finding the covering items of a class, each of which is said to cover a subset of the class. Given a class $[x]$, where $y \in [x]$, then y is said to cover the subset of $[x]$ given by $cov(y) = [x] \cap [y]$. From the covering set of a class (given in Equation (7)), the maximal cliques for that class are derived by recursively discovering the maximal cliques of each covering set item. The maximal clique set of $[x]$ is then each covering set item's maximal clique set intersected with $[x]$ and prefixed with $[x]$'s class identifier.

$$[x]_{cov} = \{y : y \in [x] \wedge cov(y) \neq \emptyset \wedge cov(y) \not\subseteq cov(z) \forall z : z \in [x] \wedge z < y\} \tag{7}$$

This process is illustrated in Figure 8, in which Figure 8(a) and Figure 8(c) present the association lattice derived from $V_2 = \{ab, ac, ae, bd, cd, ce, de\}$ and the associated prefix classes. The cliques, presented in Figure 8(c), are then found by recursively calculating cover sets for each prefix class from which the maximal clique sets in Figure 8(d) are derived.

The decision to use Eclat or Clique is based on the nature of the dataset as the enumeration of maximal cliques can be computationally expensive for dense datasets. As density increases prefix-based decomposition, Eclat becomes more competitive. An extension to Eclat (dEclat) was subsequently proposed by Zaki and Gouda [2001] that used the concept of diffsets (see Section 2.2) to reduce memory requirements by only storing the differences in the tidList of a candidate itemset from its generating itemset.

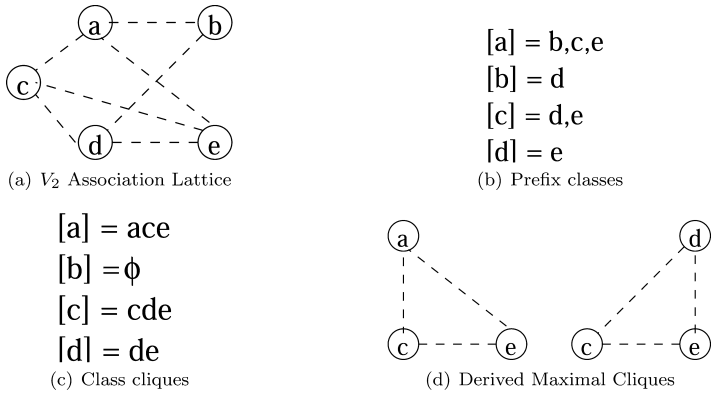


Fig. 8. Maximal clique derivation from $V_2 = \{ab, ac, ae, bd, cd, ce, de\}$.

Partial Support Tree [Goulbourne et al. 2000] requires only a single pass of a horizontally-organized dataset, during which an enumeration-set tree is dynamically constructed and *support* counts are partially calculated. *Support* calculation is then completed using DFT. This algorithm is effective for tasks in which $|E|$ is small or wher many duplicate objects exist. The tree is constructed during the dataset scan by creating new nodes in the appropriate location for each unique object. Dummy nodes are also inserted as required to maintain an enumeration-set tree structure. During this process, interim node counts are incremented, resulting in a partial support calculation of each node $\bar{\sigma}(n)$ (Equation (8)), where d_o is the *support* in D of the unique object o and the tree's ordering is lexicographic. Based on $\bar{\sigma}(n)$, it becomes possible to compute *support* $\sigma(n)$ by appending to $\bar{\sigma}(n)$ the count of n's participation within nodes lexicographically proceeding n in the enumeration-set tree by using DFT (Equation (9)).

$$\bar{\sigma}(n) = \sum d_o\{\forall o : o \subseteq n, o \textit{ follows } n\} \tag{8}$$

$$\sigma(n) = \bar{\sigma}(n) + \sum d_o\{\forall o : o \subset n, o \textit{ precedes } n\} \tag{9}$$

3.1.4. Hybrid Algorithms. Hybrid algorithms, those that combine both counting and intersection accrual strategies, are based on the premise that, although counting is more effective than intersection calculation for small κ, as κ increases, intersection techniques become more viable. This occurs as the cost of finding and accruing small κ itemsets is less in a hash-tree than the cost of calculating the intersection of large itemsets (i.e., if κ is small, then generally $|a.tidList|$ is large). However, as κ increases, hash-tree location becomes more expensive and intersection becomes faster.

Based on this premise, Agrawal and Srikant [1994] developed the first hybrid approach, AprioriHybrid, which switches from counting to intersection accrual when $|C_\kappa| << |O|$, using Apriori for counting accrual and AprioriTid for intersection accrual. This algorithm was subsequently optimized by Hipp et al. [2000b] and by Bodon [2003]. Hipp et al. eliminate the need to build a new structure on organization change. This was achieved by using a hash-tree like structure containing pointers to tidList sets instead of counters that could then be used for both accrual techniques. Bodon developed Apriori-Brave in which both delayed accrual and organization switching are based on memory usage. Moreover, the algorithm implements dataset pruning, removing invalid items from the dataset.

Dynamic Counting and Intersect (DCI) [Orlando et al. 2001a] is an extension of DCP that switches to intersection accrual from DCP's direct counting method when

- Since C_κ is lexicographically ordered, the number of intersections required is reduced by caching intermediate intersections.
- As bit vectors are relatively sparse, speedup is achieved by skipping the runs of 0's so that only vector segments containing 1's are actually intersected (similar to HBM).
- Dataset object pruning (column removal) is conducted when none of its items participate in V_κ. This is achieved by maintaining a bit vector initialized to 0's, that is, *OR*'ed with every discovered valid itemset. At the end of the scan, any item's containing a 0 are deleted.
- When the vertical layout is created, it is organized in increasing order of *support* as the higher the frequency, the greater the likelihood of the item participating in V_κ, where κ is large. Thus as the section of the dataset accessed shrinks, spatial locality is enhanced.
- During pruning, the columns are reordered to increase the length of runs of 0's and 1's to facilitate pruning based upon runs of 0's.

Fig. 9. Processing optimizations.

the dataset is able to fit into memory. The technique then transforms the dataset into vertical organization and uses a κ-way intersection of the itemset's participants to calculate itemset *support*. This was adopted to reduce memory usage as it requires the maintenance of V_1 only instead of all $V_{\kappa-1}$ itemsets which can require orders of magnitude more memory. From a processing perspective, the κ-way intersection technique is more expensive than the 2-way intersection technique. This led to the development of the set of computational optimizations, presented in Figure 9, that have been shown to be of significant benefit when many complex patterns are discovered due to inherent overheads. These optimizations are only introduced if $|C_\kappa| \gg |\bar{E}| \mid \bar{E} = E \in C_\kappa$.

kDCI [Lucchese et al. 2003] subsequently extends DCI by using density heuristics and efficient data structures in order to adapt algorithm behavior based on dataset characteristics. A novel counting inference strategy is proposed based on the concept of key patterns, discussed in Section 4.2, through which an itemset's support can be inferred, alleviating accrual of some itemsets. The effectiveness of this strategy is based on the ratio of valid itemsets to key patterns, a fact only known upon computation completion. However, the authors discovered that this effectiveness metric can be derived from the average item support (after the first dataset scan), providing early justification as to the benefit of applying counting inference.

VIPER [Shenoy et al. 2000] uses accrual to calculate V_1 and V_2, then changes to an intersection-based technique, storing itemsets in a compressed bit-vector format termed *Snakes* to minimize storage costs. Candidate generation is optimized by incorporating equivalence-class concepts, grouping together all V_κ with a common $\kappa - 1$ prefix from which $C_{\kappa+1}$ will be derived. By taking advantage of the spatial locality of itemsets, this technique allows simultaneous subset search, reducing processing time. The accrual process incorporates delayed accrual by creating a tree structure with leaf nodes V_κ and internal nodes containing supersets of these leaf itemsets. This structure is then used to propagate accrual, bottom-up, starting from the leaf items to all relevant supersets in a single dataset scan. Once complete, nodes with a count exceeding *minsup* are appended to the relevant V. The algorithm also introduces techniques to minimize the number and size of Snakes required and also deletes all Snakes created in previous scans that are no longer required for future computation.

3.2. Pattern Growth Algorithms

In contrast to the more prolific candidate generation techniques, pattern growth algorithms eliminate the need for candidate generation through the creation of complex *hyperstructures* (data storage structures used in pattern growth algorithms). The structure is comprised of two linked structures, a *pattern frame* and an *item list*, which

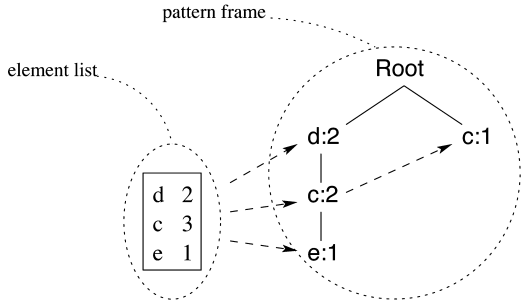


Fig. 10. Example hyperstructure.

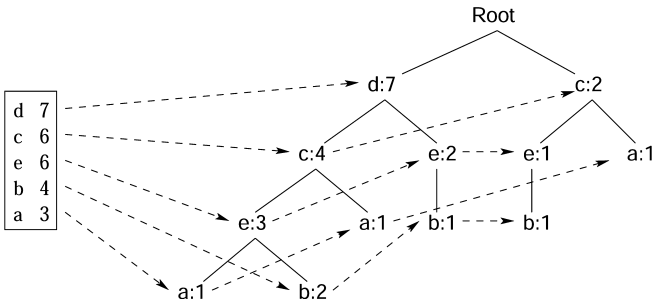


Fig. 11. FP-Growth: FP-Tree and FP-Link.

together provide a concise representation of the relevant information contained within D. The first stage of analysis populates the hyperstructure and, so long as the representation can be maintained in memory, further dataset access is not required. Subsequent mining involves DFT analysis of the pattern frame, accessed through the item list. Figure 10 presents a sample tree form of hyperstructure from the FP-growth algorithm, illustrating the linkage between the item list and pattern frame. However the nature of the hyperstructure is algorithm dependant, varying in relation to the substructures and the underlying semantics.

The fundamental pattern growth algorithm, FP-Growth, was proposed by Han and Pei [2000] and uses a tree-based pattern frame (*FP-Tree*) and an associated link structure (*FP-Link*) within the analysis process. FP-Tree construction begins with the classic BFT calculation of V_1, the contents of which are sorted in order of descending *support*. A second dataset scan then reduces and reorders each object to comply with the ordering of V_1 and is inserted into the hyperstructure (*FP-structure*), the result of which is presented in Figure 11.

Given a reduced and ordered object $\bar{o} = \{e_1, \ldots e_n\}$, its insertion into FP-structure is recursive, requiring a search and update process within FP-Tree and then an update of FP-Link for each item in $\bar{o}$. FP-Tree insertion starts with $\Re$ as the active node and proceeds by checking the object's first item (e_1) against the existing first-level FP-nodes. If e_1 does not exist, a new first-level FP-Tree node is created for e_1 with a count of 1, otherwise the existing nodes count is incremented. The item e_1, is then inserted into FP-Link in a similar manner, creating it if it does not exist and incrementing it if it does. Finally, a reference, if the FP-tree node is new, is established between it and the FP-Link entry. FP-Link thus maintains the total *support* of an item, and an FP-Tree maintains the *support* of an item in a particular sequence.

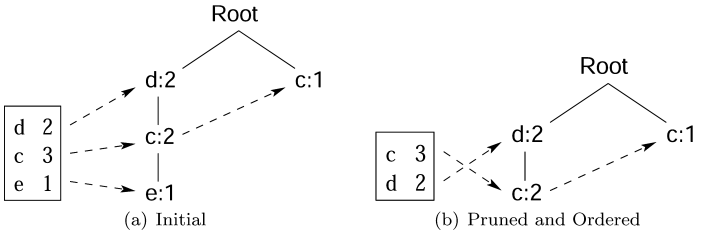


Fig. 12. a-constrained FP-structures.

The insertion recursively continues inserting each item from $\bar{o}$ into the FP-structure in a similar manner. This is achieved by removing e_1 from $\bar{o}$ and making it the active node, thereby progressing down a level of the FP-Tree structure for each item in $\bar{o}$ until $\bar{o} = \emptyset$. This process is undertaken for each reduced and ordered object, resulting in an effective representation through the common path sharing of multiple objects. Due to path sharing within FP-Tree, the FP-structure representation becomes more compressed (and therefore optimized) as dataset density increases.

The subsequent mining process recursively constrains the FP-structure until the resultant constrained FP-Tree contains a single path at which point valid itemsets are generated from the combinations of the set of participant items. Each itemset's *support* is that of the least frequent item in that combination.

Given a branched FP-Tree, a set of constrained FP-Trees are generated for each participant item, represented within FP-Link and processed from least to most frequent. For each item, a conditional pattern base is derived that contains the prefix patterns of the specific item within the current FP-Tree. This pattern base then replaces the current object set ($\bar{o}$) as the data source for the constrained FP-structure. Subsequently, the new FP-structures are built from the conditional pattern base and then pruned removing any items with insufficient *support*.

For example, the FP-Tree in Figure 11 consists of many paths and therefore conditional FP-structures are required to derived the valid itemsets. By processing the FP-Link bottom-up, the algorithm first generates the a-constrained FP-structure, or a-struct (illustrated in Figure 12(a)). The conditional pattern base of a-struct is based on a's prefix patterns in FP-Tree $\{(dce:1), (dc:1), (c:1)\}$, where the count relates to the co-occurrence of a with these itemsets within FP-Tree. After construction a-struct is pruned, removing all items with insufficient support, resulting in Figure 12(b).

As a-Tree is still branched, the constraint process repeats, starting with item d, this results in an empty conditional pattern base and the derivation of the itemset $ad : 2$. The next item, c is then processed, resulting in the conditional pattern base $\{(d : 2)\}$ and the itemset $ac : 3$. The ac-struct is then generated, resulting in ac-Tree that consists of a single node $(d : 2)$. Since ac-Tree has a single path, further constraint is not required, and valid itemsets are derived from all path combinations in union with the constraint set ac, resulting in the generation of the itemset $acd : 2$. The valid itemsets, therefore, generated in relation to a are $\{(a : 3), (ad : 3), (ac : 3), (acd : 2)\}$. The algorithm recursively continues until FP-Link has been traversed.

FP-Growth has been shown to be effective in the mining of dense datasets as the FP-structure concisely encapsulates valid item information. However, as the number of different items, $|E|$, increases, the size of FP-Tree typically expands exponentially due to the reduced sharing of common prefixes. To address this, FP-growth* was proposed by Grahne and Zhu [2003a]. It uses an additional array-based structure to reduce the number of tree traversals required during analysis. This array-based structure saves on general traversal times and enables the direct initialization of the next level of

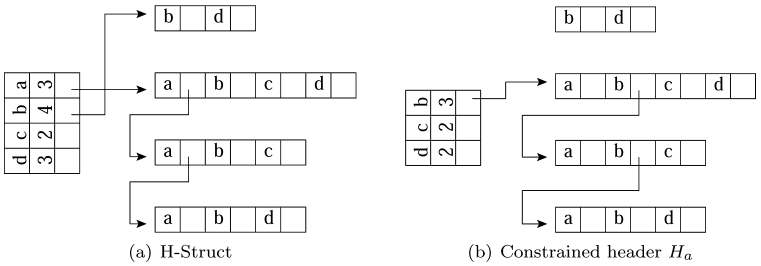


Fig. 13. H-Mine data structure.

FP-Trees. However, its instantiation is generally not warranted in dense datasets or in the first levels of recursively constructed FP-Trees from sparse datasets since they are based on the most common prefixes available. A density heuristic was therefore devised to determine the benefit of constructing the array.

Wang et al.[2002] proposed top-down variation of FP-Growth (TD-FP-Growth) that alleviates the need to generate conditional pattern bases and physical projections of the trie. The algorithm constructs the trie in a similar fashion to FP-Growth, however, it then processes the FP-List top-down, mining through the recursive creation of conditional FP-Lists, which all refer to the global FP-Tree.

H-Mine [Pei et al. 2001] extends the pattern growth concepts introduced in FP-Growth, although an array-based hyperstructure is used. Population of the hyperstructure occurs in a similar manner to FP-Growth with the first scan identifying V_1, and the second creating the H-struct hyperstructure from it. However, if the constructed H-struct cannot fit into memory, D is partitioned, in a similar manner to the Partition Algorithm, each of which is individually analyzed and consolidated.

The two parts of H-struct are the linked projected object set P and header table H, relating to the pattern frame and item list, respectively. Each $p \in P$ contains the valid items of an object, derived from V_1. The header table H contains a list of the valid items F and their counts. During construction, all P that begin with the same item are linked together in a queue, the head of which resides in the header table H (Figure 13(a)).

Analysis is conducted through a traversal of H creating conditional H-structs for each header item that is a queue head within P. This process is recursive, progressively creating further constrained H-struct until it is empty. This process is effectively the same as the DFT of an enumeration-set structure, where all V_κ are identified at that level of constraint through analysis of the header table. This analysis process is illustrated in Figure 13, with Figure 13(a) illustrating a populated initial H-struct and Figure 13(b) presenting its constraint in relation to item a.

The advantage of H-Mine over FP-Growth is apparent as the same pattern frame structure is used with semantics changed through pointer manipulation, rather than the creation of new constrained FP-structures required in FP-Growth. Figure 13(b) presents a-struct from which valid itemsets containing a of length 2 are represented in H, resulting in $ab : 3, ac : 2, ad : 2$. Analysis proceeds until all of H has been processed, resulting in the identification of V.

ITL-Mine [Gopalan and Sucahyo 2002] optimizes H-Mine by requiring only a single dataset scan, reducing I/O, and through the maintenance of a static set of links in the hyperstructure. The single scan of D creates the underlying structures that are similar to H-struct, except that the header tables maintain all items not just those that occur first in any $p \in P$. This additional linkage avoids the progressive recalculation of linkages during processing that is apparent in H-Mine. After the first pass, V_1 is derived

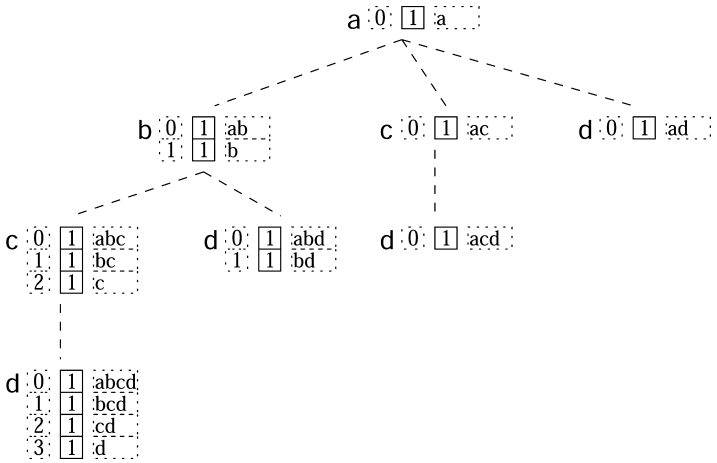


Fig. 14. CT-ITL compressed tree structure.

from the header counts, and the associated list structure is reduced by removing the invalid items.

Subsequent analysis is recursive, for $\kappa = 2$ the analysis is prefix-based, involving a traversal of P, during which a temporary structure, *ITL-List*, is created and subsequently used in $\kappa > 2$ analysis. ITL-List initially contains those items that co-occur with the specified prefix item, their co-occurrence count, and relevant tidList. After generating V_2, the tidLists of each itemset are available in the temporary structure and hence, $V_\kappa \mid \kappa > 2$, that extend the designated prefix can be recursively generated using intersection and ITL-List.

CT-ITL [Sucahyo and Gopalan 2003] extends ITL-Mine by using a compressed pattern framework structure to reduce storage space and traversal overheads. Although requiring two dataset scans, it has been shown to be more scalable than ITL-Mine as dataset size increases. Based on a reordered header structure built during the first pass, the algorithm creates a compressed prefix tree, containing all possible itemsets, based on V_1. This structure compresses the regular prefix tree structure (Figure 3(b)) by storing identical subtrees within a single set of nodes and storing the additional required information within a node associated array.

Given $E = abcd$, Figure 14 illustrates the resulting prefix structure. The figure's array structure (annotated with dashed structures to facilitate understanding) shows that each node array item maintains the count for a unique itemset with the array index specifying the tree level at which accrual for that particular itemset begins. For example, given the node abc, the array consists of three items indicating the counts for the itemsets abc, bc, and c, respectively. The resulting compressed prefix structure is then populated by scanning D and inserting all objects which have been pruned so that all invalid items are removed, resulting in $\bar{D}$. The tree then represents an object summary that is used to create a similarly compressed projected object list structure.

Instead of representing each object as a row within P as in H-Mine, compressed ITL, derived from the compressed prefix-tree structure, consists of the set of Maximal Object Sets (MOS) that encapsulate the required object information. In a similar manner to the compressed tree, an associated array maintains each MOS's subset count, thereby maintaining the count of each unique object. In this way, the array index specifies the MOS item from which accrual occurs in order to produce the itemset to which the specific count refers. For example, given the MOS abd, the count associated with the second array item refers to $|o| \mid o = \{b, d\} \wedge o \in \bar{D}$. Subsequent mining is recursive

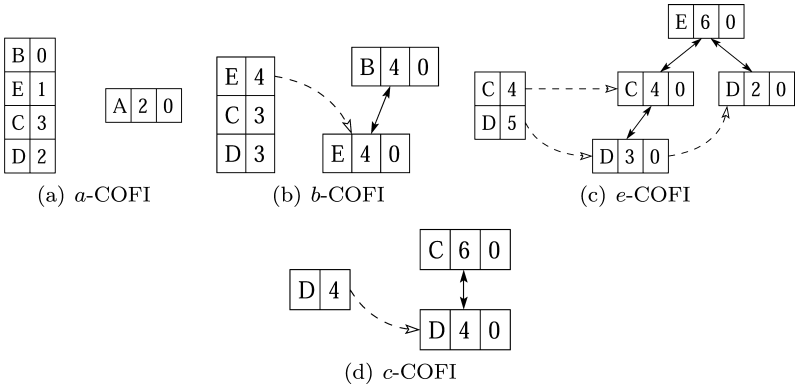


Fig. 15. COFI trees based on FP-Tree in Figure 11.

and similar to ITL-mine, although the temporary structure used and the intersection method are slightly modified to accommodate summarized objects.

Opportunistic Projection [Liu et al. 2002] is a hybrid pattern growth algorithm that extends the authors prior work, namely FP-Growth and H-Mine. Mining occurs through the construction of *FIST*, a prefix tree with associated counts. The algorithm constructs the tree using classical BFT and dataset reduction techniques until the data structures required for further mining can be held in memory from which point memory resident projection-based techniques are used.

Processing is based on FP-Tree or H-struct, depending on the density of the projected transaction set *PTS*. For small κ, the *PTS* items have less chance of prefix sharing and hence relative *support* and density is low however as κ increases, the size of *PTS* decreases, and relative support increases. As noted by the authors, FP-Tree compresses poorly in low relative support situations and therefore is used for high κ, while H-struct, which has a linear growth, is a better choice for small κ. *PTS* items reduce quickly for small κ and therefore H-mine filtering is incorporated when using H-struct. However, *PTS* reduction slows as κ increases, reducing the filtering benefit and making the creation of conditional FP-Trees unwarranted. To reduce FP-Tree filtering costs, the algorithm introduces the concept of Pseudo Projections, whereby the reduction factor can be determined and only if warranted will a conditional FP-Tree be created.

COFI proposed by Osmar and El-Hajj [2003] uses the concept of Co-Occurrence Frequent Item trees to provide an efficient pattern growth algorithm that uses a top-down nonrecursive technique. The algorithm first constructs structures similar to FP-Growth, except that the FP-Tree is double-linked allowing for fast traversal without recursion. Mining proceeds through the construction and analysis of COFI trees, each of which is based on a valid FP-Link item (processed in order of ascending frequency). Each COFI tree is then mined independently without recursively building further constrained subtrees.

A COFI tree, based on an item x, is constructed by traversing FP-Tree for each item occurring after x in FP-Link, finding their level of co-occurrence with x. If locally valid with respect to a, the item is subsequently used in the construction of x-COFI. Given the FP-Tree illustrated in Figure 11, subsequent analysis (given a support of 4) results in the COFI trees presented in Figure 15 (from which the similarity to constrained FP-Trees is evident). Each COFI node consists of an item occurrence in relation to the base item and a participatory count used in subsequent mining.

Each COFI tree is sequentially constructed, mined if locally valid items exist, and discarded before the invocation of the next COFI tree, minimizing memory usage. This

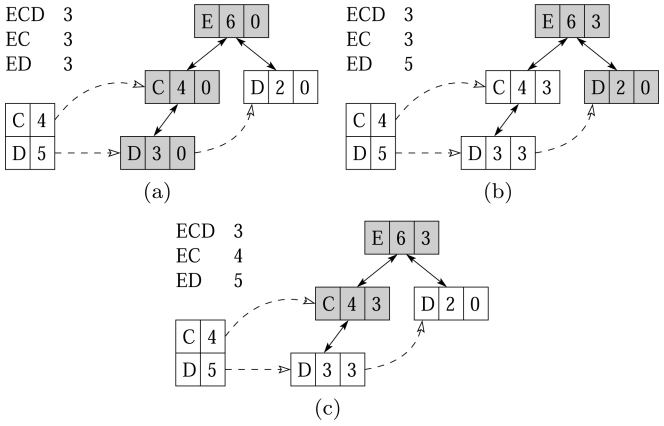


Fig. 16. COFI tree mining based on *e*-COFI in Figure 14.

mining process is discussed with reference to Figure 16, which is based on *e*-COFI (Figure 15(c)). Processing occurs in descending FP-Link frequency order where each item's link list is traversed and each participant branch analyzed separately. The first step (Figure 16(a)), therefore, involves analysis of the branch *ECD* from item *D*, where the frequency of a branch is the frequency value less the participation of the specific item, therefore the frequency of *ECD* is 3. The participation values of all nodes in the branch are incremented by 3, and all subpatterns of the branch are generated with the same frequency value. In Figure 16(b) the branch *ED* is analyzed. As this itemset already exists, its count is consequently updated ($ED : 5$), and the participation count of *D* in branch *ED* updated. Figure 16(c) presents the analysis of branch *EC* which has a frequency value of 1 and already exists, resulting in $EC = 4$. *e*-COFI is then complete, resulting in the generation of the itemsets $\{ECD : 3, EC : 4\, ED : 5\}$, and the next COFI tree can then be constructed and mined. COFI results in the accurate generation of all valid itemsets, while using less memory than FP-Growth due to its nonrecursive analysis process.

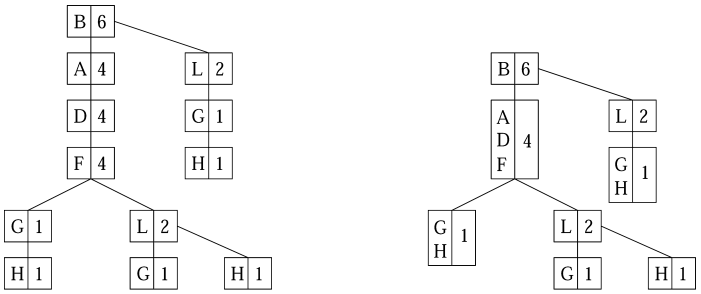
PatriciaMine [Pietracaprina and Zandolin 2003] proposes the use of a compressed trie, known as a Patricia trie, to alleviate the need to swap between trie and array-based data structures based upon dataset density as proposed in H-mine and Opportunistic Projection. Furthermore, it is claimed that the compression technique used results in a structure comparable (in terms of size) with array-based structures in the presence of sparse datasets.

A Patricia trie is a modification of a regular trie in which each maximal chain of nodes that have a common count (support) are coalesced into a single node that inherits the count and stores the items in the same sequence. Figure 17 illustrates this through a common dataset representation using both a regular and Patricia trie. The implementation of PatriciaMine also incorporates the constraint of physical projections in a similar manner to Opportunistic Projection and uses novel mechanisms to directly generate groups of itemsets supported by the same transaction subset.

Table I summarizes the classic Algorithms presented in this section.

4. CONDENSED REPRESENTATION ALGORITHMS

Condensed representation algorithms provide a generic optimization to fundamental algorithms by producing a reduced result set from which all valid itemsets can be



(a) Regular Trie (b) Patricia Trie

Fig. 17. Patricia Trie example from Pietracaprina and Zandolin [2003].

derived. There are four techniques used to produce condensed representations: closed sets, counting inference, deduction rules, and freesets.

4.1. Closed Itemset Algorithms

Closed itemset algorithms identify a subset of valid itemsets from which all valid itemsets can be derived without further mining. The theoretical foundation of closed itemset algorithms is based on the closure of the Gauloise connection [Ganter and Wille 1999] in which a closed pattern is the largest pattern common to a set of objects within D. All nonclosed patterns have the same critical properties (in this case *support*) as its closure, which is the smallest closed pattern containing it. The closure of an itemset a, denoted $cl(a)$, is then the smallest closed pattern containing a. Given that $g(o)$ is the set of items common to all objects in $o \mid o \subset O$ and $f(a)$ is the tidList of all items a. The closure of any itemset a is found by constructing the set of objects in which the itemset a appears $f(a)$ and from this, calculating the set of items $\bar{a}$ that are common to all objects in $f(a)$ by applying $g(f(a))$. From this, $\bar{a}$ must be the closure of a as $g(f(a)) = \bar{a} = cl(a)$ as it identifies the set of items that always occur with a, hence $\bar{a} \supseteq a$ and $\sigma(\bar{a}) = \sigma(a)$.

The following algorithms discover the closed set of valid itemsets Cl within D, using a variety of techniques from which V can be derived. The inclusion of closure constraints optimize analysis through search space reduction, particularly for highly correlated datasets.

The derivation of V from Cl is simple, however, since Cl implies V, the generation and presentation of closed rules may facilitate user interpretation due to the reduced number. Closed rules [Pasquier et al. 1999b] are a reduced set of inferences derived from C, where, given a closed itemset a, an inference r is of the form $x \Rightarrow a - x \mid x \subset a \ \wedge \ a \in Cl \ \wedge \ x \in Cl$ the confidence of which, $\gamma = (\sigma(a)/\sigma(x))$, is available.

A-Close [Pasquier et al. 1999b] uses a candidate generation BFT approach consisting of two stages, generator set discovery and closed set derivation. The generator set (G) is constructed in regular Apriori fashion, however, it incorporates additional pruning such that, if a $G^i_{\kappa+1}$ has the same *support* as any of its κ subsets, then, based on the Gauloise connection, it has the same closure and is removed from $G_{\kappa+1}$, resulting in G containing all locally closed sets. The second stage calculates the global closure of each generator by performing an intersection of all objects in which the generator occurs. Figure 18 illustrates the identification of G_1 and G_2 with pruned itemsets due to infrequency and subset *support* equivalency crossed out. The rightmost table presents the final closed set Cl that identifies all global closed itemsets.

Table I. Summary of Classic Algorithms

Name	Author	Year	Org.	Structure	Scans	Contribution
Candidate generation algorithms: Merge based or BFT						
AIS	Agrawal	1993	HL	—	k	First association mining algorithm
SETM	Houtsma & Swami	1993	HL	—	k	Merge & count
Apriori	Agrawal & Srikant	1994	HL	Hashtree	k	DCP
OCD	Mannila et al.	1994	HL	Hashtree	k	Merge operators
AprioriTID	Agrawal & Srikant	1994	VL	Hashtree	1	TID
Apriori *ext*	Agrawal & Srikant	1994	HL	Hashtree	$\leq k$	Delayed accrual
Partition	Savasere et al.	1995	VL	Hashtree	≤ 2	Partitioning
SPINC	Mueller	1995	—	—	≤ 2	Partitioning & Incremental candidate construction optimisation
DLG	Yen & Chen	1996	VV	Graph	$\leq k$	Association graph
DIC	Brin et al.	1997	HL	Hashtree	$\leq k$	Partitioning & checkpoints
DHP	Park et al.	1997	HL	Hashtree	k	Possible frequent itemsets & delayed accrual
AS-CPA	Lin & Dunham	1998	—	—	≤ 2	Partitioning & Anti-skew
NBM	Gardarin et al.	1998	VV	—	k	Indexing
HBM	Gardarin et al.	1998	VV	—	k	Second level indexing
ColumnWise	Dunkel & Soparker	1999	HV	—	k	Column based intersections
PHP	Ozel & Guvenir	2001	HL	Hashtree	k	Perfect hashing
DCP	Orlando et al.	2001	HL	Hyper	k	Dataset pruning
Candidate generation algorithms: Extension based or DFT						
Tree Projection	Agrawal et al.	1999	HL	Trie	1	Projection, matrix accrual & BFT and Hybrid versions.
Apriori-df	Pijls	1999	VV	Trie	$V_1 - 1$	DFT trie construction
Depth Project	Agrawal et al.	2000	VV	Trie	1	Novel accrual techniques
Eclat	Zaki	2000	VV	Lattice	2	Prefix-based equivalence relations
Clique	Zaki	2000	VV	Lattice	2	Maximal cliques
AprClique	Zaki	2000	HV	Lattice	2	Maximal cliques
Part. Sup. Tree	Goulbourne et al.	2000	HL	Trie	1	Novel accrual
dEclat	Zaki & Gouda	2001	VV	Lattice	2	Inclusion of diffsets
Candidate generation algorithms: Hybrid						
AprioriHybrid	Agrawal & Srikant	1994	HL⇒VL	Hashtree	$< k$	
Hybrid	Hipp et al.	2000	HL	Hashtree	$< k$	Improved strategy switching
VIPER	Shenoy et al.	2000	HL⇒VV	Hyper	$< k$	Compression
DCI	Orlando et al.	2001	HL⇒VV	Hyper	$< k$	Reduced memory usage
kDCI	Lucchese et al.	2003	HL⇒VV	Hyper	$< k$	Adaptive algorithm behaviour
Apriori-Brave	Bodon	2003	HL⇒VL	Trie	$< k$	Reduced Storage & memory management
Pattern Growth Algorithms						
FP-Growth	Han & Pe	2000	HL	Hyper	2	Trie based
H-Mine	Pei et al.	2001	HL	Hyper	2	Array based & partitioning
TD-FP-Growth	Wang et al.	2002	HL	Hyper	2	Top down processing
ITL-Mine	Gopalan & Sucahyo	2002	HL	Hyper	1	
Opp. Projection	Liu et al.	2002	HL	Hyper	2	Hybrid: FP-growth & H-Mine
FP-Growth*	Grahne & Zhu	2003	HL	Hyper	2	Traversal reduction for sparse datasets
CT-ITL	Gopalan & Sucahyo	2003	HL	Hyper	2	Compressed structures
COFI	Osmar & El-Hajj	2003	HL	Hyper	2	Co-occurence Frequent Itemset trees
PatriciaMine	Pietracaprina & Zandolin	2003	HL	Hyper	2	Patricia trie

An earlier algorithm, Close [Pasquier et al. 1999c], identified the closed itemset for each generator as it is discovered instead of separating out global identification. A-Close is reported as more efficient on dense datasets.

CLOSET [Pei et al. 2000] is based on pattern growth concepts and uses similar data structures and processes as described in FP-Growth, although it introduces additional pruning techniques to discover only the set of closed itemsets. This subsumption

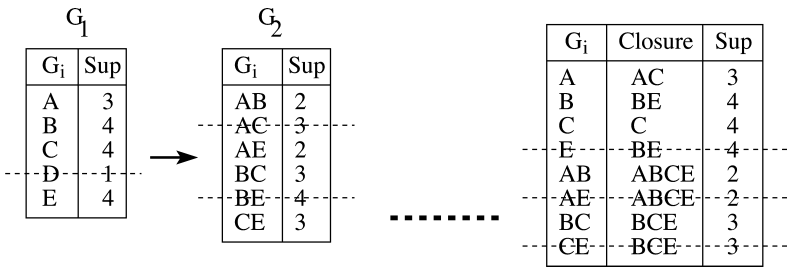


Fig. 18. A-Close Process diagram (Pasquier et al. 1999b).

testing ensures that a new valid itemset is only appended to Cl if it is not subsumed by an existing member, while candidates cannot subsume existing members due to the bottom-up order of processing implemented within FP-Growth. Furthermore, CLOSET facilitates closed set discovery through the inclusion of two process reduction techniques that result in search space reduction based on closed set characteristics.

—Given an (a)-constrained dataset D_a, if an item or itemset, b, appears in all D_a objects, then the closure of $a \cup b$ will subsume the closure of a. Therefore, b is removed from D_a and appended to the constraint, therefore D_a becomes $D_{a \cup b}$.

—Given a valid closed set a and the subsequent discovery of a closed itemset b such that a subsumes b, then D_b is not processed as any resultant closed itemsets will be subsumed by a.

Subsequent extensions to CLOSET have been proposed in CLOSET+ [Wang et al. 2003] and FPClose [Grahne and Zhu 2003a]. CLOSET+ introduces a hybrid tree projection method, incorporates item skipping, and proposes two efficient subset-checking structures. The hybrid tree projection method proposes the use of a top-down pseudo-projection technique for sparse datasets, through which the FP-Tree is not physically replicated for each conditional base. Item skipping prunes the search space by recognizing that, if an item has the same *support* in different conditional FP-Lists, then that item can be removed from consideration in the higher level FP-Lists. FPClose extends CLOSET through the creation of a local Closed Frequent Set trees (CFI-tree) for each conditional FP-Tree; instead of performing global subset checking, local checking is performed first, significantly reducing subset testing in large datasets (see also the discussion of FPMax* in Section 5.1).

(1) If $o(a) = o(b) \mid a \cap b = \emptyset$, then $cl(a) = cl(b) = cl(a \cup b)$. This implies that all occurrences of a can be replaced with $a \cup b$, and b can be removed from further consideration as its closure $cl(b) = cl(a \cup b)$.

(2) If $o(a) \subset o(b)$, then $cl(a) \neq cl(b)$ but $cl(a) = cl(a \cup b)$ implies similar replacement however, as $cl(b) \neq cl(a)$, b cannot be removed from further consideration.

(3) If $o(a) \supset o(b)$, then $cl(a) \neq cl(b)$ but $cl(b) = cl(a \cup b)$, the inverse of Property (2) is true in which b is replaced by $a \cup b$ instead of a, but a is not removed from consideration.

(4) If $o(a) \neq o(b)$, then $cl(a) \neq cl(b) \neq cl(a \cup b)$ indicates that both a and b lead to different closures and hence no replacement occurs.

CHARM [Zaki and Hsiao 2002] is similar to CLOSET in its general approach, but uses an enumeration-set tree and the concept of equivalence classes introduced in Eclat (Section 3.1.3) where two itemsets belong to the same κ equivalence class if they share a common $|\kappa|$ prefix. CHARM extends Eclat to discover only closed itemsets through the introduction of a set of closure properties that are presented previously where $o(a)$

$$\begin{aligned}
\sigma(abc) &\leq \sigma(ab) + \sigma(bc) + \sigma(ac) + \sigma(a) - \sigma(b) - \sigma(c) + \sigma(\emptyset) \\
\sigma(abc) &\geq \sigma(ab) + \sigma(ac) - \sigma(a) \\
\sigma(abc) &\geq \sigma(ab) + \sigma(bc) - \sigma(b) \\
\sigma(abc) &\geq \sigma(ac) + \sigma(bc) - \sigma(c) \\
\sigma(abc) &\leq \sigma(ab) \\
\sigma(abc) &\leq \sigma(bc) \\
\sigma(abc) &\leq \sigma(ac) \\
\sigma(abc) &\geq 0
\end{aligned}$$

Fig. 19. Bounding rules to derive the support of abc.

indicates the set of objects containing itemset a. While properties (1) and (2) result in equivalence class reduction and hence facilitate closed-set convergence, Properties (3) and (4) result in new equivalence class information that generally requires additional processing. The inclusion of these properties within Eclat-based processing occurs after the discovery of a new valid itemset a, whereby the closure properties are then applied to control a's influence on the generation of the closed-set equivalence class lattice.

4.2. Counting Inference

Counting inference [Bastide et al. 2000] (implemented as PASCAL, an Apriori extension) is based on the observation that itemsets can be considered equivalent if they are included in the same set of objects, referred to as a class. Mining can then be reduced to the identification and validation (*support*) of the set of *Key Patterns*, where a *Key Pattern* is a minimal itemset within a class from which the validity (*support*) of all other itemsets can be inferred. A pattern can only be a *Key Pattern* if all its subsets are *Key Patterns*, that is, no subset belongs to the same equivalence class. This theory leads to two analysis optimizations, especially in highly correlated datasets.

—Only the Key Patterns need to be considered during each dataset scan as nonkey itemset support is equal to the minimum support of its $\kappa - 1$ subsets.

—The number of dataset scans will often be reduced because the set of Key Patterns is often found before the last iteration because, if a single $\kappa - 1$ subset of a candidate itemset is nonkey, then the itemset is nonkey; hence, from a certain level, all itemsets may be known to be nonkey and therefore can be derived from subsets without requiring another dataset scan.

4.3. Deduction Rules

Calders and Goethals [2002] concisely represent the set of valid itemsets through a set of deduction rules using the mathematical inclusion-exclusion principle. The rules are used to derive tight bounds on the *support* of an itemset given that the *support* of its subsets are known. Given an itemset $i \mid j \subseteq i \subseteq I \ \wedge \ j \neq \emptyset$, its upper and lower bounds are derived using Eq. (10) if $|i \setminus j|$ is odd, and Equation (11) if even. For example, Figure 19 provides all the possible bounding rules to derive the itemset abc. The concise representation is therefore a set of valid itemsets whose *support* cannot be derived, the set of nonderivable itemsets, from the deductive rules.

$$\sigma(i) \leq \sum_{j \subseteq x \subset i} (-1)^{|i \setminus x|+1} \sigma(x) \tag{10}$$

$$\sigma(i) \geq \sum_{j \subseteq x \subset i} (-1)^{|i \setminus x|+1} \sigma(x) \tag{11}$$

The implementation used extends Apriori, incorporating deductive rules to eliminate from consideration those itemsets whose *support* can be deduced, resulting in less mining time and a more concise result set of all nonderivable itemsets. From this result set, the *support* of all other valid itemsets can be found by using deductive rules.

4.4. Free Sets

Disjunction free sets [Bykowski and Rigotti 2001] are condensed representations based on the theory of disjunction. A disjunctive rule is founded on the concept that given itemset $A = \{abcd\}$, then $\sigma(A)$ can be derived from $\sigma(ab)$, $\sigma(a, b, c)$, and $\sigma(abd)$ as the sum of $\sigma(A)$ and $\sigma(ab)$ is equal to the sum of $\sigma(abc)$ and $\sigma(abd)$, given that ab does not exist in D without either c or d. The calculation of $\sigma(a)$ can therefore be derived from the support of these particular subsets, given the presence constraint without scanning D and is referred to as a nondisjunction free set since it can be calculated from disjunctions. Hence mining is reduced to discovery of the set of disjunction free-sets within the search space where disjunction free sets also display nonmonotonicity in that, if an itemset is not disjunction free, then no superset of it will be either.

By using the discovered disjunction free sets in conjunction with the negative border set, which consists of the smallest itemsets that are not valid disjunction free sets, the validity of any itemset can be derived without scanning the dataset. The authors implement both DFT and BFT implementations of the algorithm and claim that it is superior to closed-sets.

δ-Freesets [Boulicant et al. 2001] provide condensed representation from which the *support* of any valid itemset can be closely approximated. The level of approximation is shown to be acceptable for many mining tasks and the trade-off against mining time advantageous. A δ-Freeset is an itemset such that its participant items cannot be used to form a strong rule where strength is a user-defined metric (δ). For example, given itemset abc, if $ab \Rightarrow c$ is a strong rule (it holds with less than δ exceptions), then $\sigma(abc)$ can be approximated using $\sigma(ab)$. δ-Freesets are also antimonotonic, that is, if an itemset A is not a δ-Freeset, then B is not either given that $B \supset A$.

The implementation, MineEx, is an Apriori-based extension, constraining results to those itemsets that are δ-Free. Once analysis is complete, any itemset's validity can be approximated by finding the smallest support among its valid δ-free subsets.

5. INCOMPLETE SET ALGORITHMS

Incomplete set algorithms are a variation of condensed representations whereby a reduced result set is produced that can provide useful (although incomplete) information about dataset inferences, reducing analysis by discovering incomplete information about the complete set of valid itemsets within D. There are two significant types of incomplete set algorithms: sampling and maximal valid sets. Sampling algorithms analyze only a portion of the dataset and Maximal Frequent Set algorithms (MFS) identify only those valid itemsets for which no valid supersets exist. Sampling works by reducing $|D|$, and MFS works through the introduction of new pruning strategies[1].

[1]While MFS relates to the inclusion of the *support* heuristic (Frequent), the pruning strategies employed are generically applicable to any analysis that incorporates nonmonotonic heuristics. However, due to general usage, this algorithmic group will be referred to as MFS instead of maximal valid sets which is more correct given the broader range of heuristics that can actually be applied.

5.1. Maximal Frequent Set algorithms

Maximal Frequent Set (MFS) algorithms identify all itemsets within D for which no valid supersets exist. Therefore, although the result set implies all valid itemsets through the identification of the search space boundary, the actual *support* and hence inference strength of internal itemsets (those within the search space boundary) remain unknown. The implementation of MFS algorithms result in reduced analysis time because MFS properties enable the inclusion of additional pruning strategies that facilitate search space reduction.

—*Upward Closure Principle (UCP)* [Bayardo Jr 1998]. Given a valid itemset A with a set of possible extensions B, if the itemset $C \mid C = A \cup B$ is valid, then C becomes the *terminal* MFS of A and no further exploration of A's supersets is required.

—*Superset Checking* [Bayardo Jr 1998] (An extension of UCP that avoids direct counting of A). If A is subsumed by or is a subset of an existing MFS, then no further exploration of A's supersets is required.

—*Parent Equivalence Pruning* [Burdick et al. 2001]. Given an extension-based analysis algorithm incorporating dataset projection, then for each child itemset generated in the enumeration-set tree, the projected object set is compared with that of its parent. If they match, the child can replace the parent node.

Maxminer [Bayardo Jr 1998] extends Apriori through the inclusion of UCP and subsumption testing to derive the set of maximal frequent sets. Furthermore, MaxMiner is underpinned by an an enumeration-set tree instead of a hash-tree within which items are dynamically sorted in order of increasing *support*.

Pincer Search [Lin and Kedem 1998] also extends Apriori through the introduction of a bidirectional search that has proven efficient in the discovery of long maximal valid itemsets. The algorithm uses a typical Apriori bottom-up search but extends it by incorporating additional pruning through the use of a Maximal Frequent Candidate Sets (MFCS) that approaches the valid search space border and hence the discovery of MFS from the top-down.

Given that initially MFCS contains a single itemset of the union of all items, analysis proceeds by iteratively generating the next bottom-up valid set V_κ, refining MFCS and then pruning V_κ, or $V_{\kappa+1}$ candidates, based on those members of MFCS determined valid. For example, given $E = \{a, b, c, d, e\}$ and $V_1 = \{a, b, d, e\}$, then MFCS is refined, ensuring that no member of MFCS is a superset of an invalid itemset which is provided by $\bar{V}_\kappa \mid \bar{V}_\kappa = C_\kappa - V_\kappa$. Thus $MFCS_1 = \{abde\}$ as item c is determined invalid. The refined members of MFCS are then validated, those found valid are MFS and all subsets within $C_{\kappa+1}$ are removed from further consideration. Thus if $\{abde\}$ is found valid, then through pruning $V_1 = \emptyset$, and $\{abde\}$ is identified as the only MFS.

This process of MFCS refinement and $C_{\kappa+1}$ pruning greatly reduces $|C|$ and facilitates MFS convergence, however, in some cases, it can be preemptive and recovery of some candidates may be required. The recovery process is achieved by creating additional $C_{\kappa+1}$'s for each MFCS member where it is a superset of the $\kappa - 1$ prefix of a V_κ^i. Given this, a new candidate is created by merging each item in the MFCS that occurs after the last item of the $\kappa - 1$ prefix with the $\kappa - 1$ prefix. Once recovery has been undertaken, regular Pincer Search analysis continues with the derivation of $V_{\kappa+1}$.

Max-Eclat and Max-Clique [Zaki et al. 1997; Zaki 2000] are based on the Eclat and Clique algorithms (Section 3.1.3), however, by using either a DFT or hybrid, DFT-BFT, approach, all MFS are efficiently identified. Using DFT, processing of each decomposed lattice begins with the identified potential maximal itemset whose validity (*support*) is determined from tidList intersection. If valid, the processing of that lattice is complete, otherwise each lattice subset is checked at the next level of decomposition until all

maximal valid itemsets have been identified. The benefit of Max-Clique over Max-Eclat is that the identified potential maximal itemsets are more refined as maximal cliques identify smaller sublattices than the equivalence class method used within Eclat. Hence less traversal is often required to discover all MFS using Max-Clique.

The Hybrid traversal search is based on the concept that the greater the *support* of a valid itemset, the more likely it is to be part of a longer valid itemset. The sublattice itemsets are placed in descending frequency order, the hybrid phase then intersects the itemsets individually, stopping when an extension becomes invalid, identifying that sublattice's MFS. The benefit of this technique is that each extension only requires a two-way Tidlist intersection. However, in both techniques, the search space is not global and some of the discovered MFS may not be globally maximal; hence some postprocessing may be required.

All-MFS [Gunopulos et al. 1997] performs random walks within an enumeration-set tree to discover MFS. A walk starts with an initial itemset and subsequently extends, using random item selection, from those items that lexicographically occur after the previously selected (extension) item. Each extension is subsequently validated using Tidlist intersection and, if invalid, a MFS has been discovered.

Analysis proceeds by deriving V_1 and removing from D all invalid items. After a specified number of walks, All-MFS calculates the set of Minimal Orthogonal items (MOE) or those items that are not subsets of discovered MFS. MOE's are then used as the starting points for a set of random walks. Although All-MFS cannot guarantee the discovery of all MFS, it does find a large portion of them.

Mafia [Burdick et al. 2001] extends DepthProject [Agrawal et al. 2000] (see Section 3.1.3) through the use of *Superset Checking* and *Parent Equivalency Pruning* to discover the set of MFI's, while GenMax [Gouda and Zaki 2001] takes a novel approach to maximality testing denoted *progressive focusing* where valid itemsets are first tested against a locally maintained set of MFI's (LMFI). Most nonmaximal valid itemsets are discovered using the local testing therefore reducing the number of subset tests required. GenMax also uses diffsets (See Section 2.2) which become more effective as density increases.

FPMax [Grahne and Zhu 2003b] provides an MFI version of FP-Growth which uses superset checking to construct an auxiliary MFI-Tree. The algorithm proceeds in a similar fashion to FP-Growth, beginning with the initial construction of a FP-Tree and FP-List. However subsequent conditional FP-Tree processing is only conducted where the conditional head, together with all valid items in the head-conditional pattern base (tail), is not a subset of an existing MFI stored within MFI-Tree. If it is not subsumed, the conditional FP-Tree is constructed and, if only consisting of a single path, is appended to MFI-Tree. Analysis proceeds as for FP-Tree, incorporating superset checking optimizations based on FP-structure and processing idiosyncrasies with the final set of MFI's represented in the MFI-Tree.

FPMax* [Grahne and Zhu 2003a] extends FPMax through the inclusion of progressive focusing (as introduced in GenMax), resulting in the maintenance of local MFI-Tree's and reducing subset testing in large datasets. Additional subset testing and processing optimizations are implemented based on the nature of the conditional-base *tail*.

5.2. Sampling-Based Algorithms

Sampling-based algorithms are based on the premise that approximate answers often suffice and therefore adequate answers can be obtained by mining a lossy compressed representation of the data. The sampling approach addresses the issue of scalability by only analyzing a representative subset of D. Work in the associated field of approximate aggregation [Gibbons 2001; Choudhuri et al. 2001] has shown that the benefits of

sampling are improved when the sample selection is guided by the user, hence improving its relevancy to the current problem.

The main issue in developing sampling techniques is to maximize the extent to which the sample reflects the generic characteristics of D while maintaining efficiency through sample size constraint. A trade-off is therefore apparent between accuracy and efficiency in naive sampling, a formal discussion of which is presented by Kivinen and Mannila [1994]. Both OCD [Mannila et al. 1994] and AS-CPA [Lin and Dunham 1998] (see Section 3.1) propose naive sampling extensions to their classic algorithms. However, the following algorithms go further by attempting to reduce the errors resulting from naive sample selection.

Toivonen [1996] introduces a sample-based analysis technique that only requires a single scan of D to discover all itemsets. The algorithm first uses a random sample and a reduced validity threshold in an attempt to calculate the superset of valid itemsets within D. Based on this, the subsequent completion of the D scan is used to discover the true set of valid itemsets.

The FAST sampling algorithm [Chen et al. 2002] attempts to reduce sampling errors by introducing a two-stage sampling technique. The first stage quickly estimates the validity of each item within D through the use of a large initial sample. The second stage derives from this a smaller sample set within which each item's validity closely represents its validity within D. By calculating item validity based on a large sample size in the first stage, it is suggested that the derived valid itemsets in the second stage will be close to the actual set of valid itemsets.

Table II summarizes the condensed representation and incomplete set algorithms presented in this section.

6. INFERENCE GENERATION

Inferences are derived (or generated) from the set of identified valid itemsets. For each valid itemset (v), a set of inferences R are produced, derived from the permutations of v, where an inference $r \in R$ is of the form $i \Rightarrow v \setminus i \mid i \subset v \wedge i \neq \emptyset$, where i is the set of antecedents and $i \setminus v$ is the set of consequents. Each inference has an associated quality heuristic confidence (γ) that is derived from the inference's strength, $\gamma(R) = \frac{\sigma(v)}{\sigma(i)}$; if this strength exceeds a specified confidence threshold (*minconf*), it is considered a valid inference and appended to the result set.

Agrawal and Srikant [1994] propose an optimization to this technique, FastGenRules, that can potentially reduce the number of itemset permutations considered. This is based on the observation that, given the derivation from v of an inference $i \Rightarrow j$ that does not meet the confidence threshold *minconf*, then all subsets of i need not be considered for generating inferences using v. For example, if $\gamma(ab \Rightarrow cd) < minconf$, then $\gamma(a \Rightarrow bcd) < minconf$ and $\gamma(b \Rightarrow acd) < minconf$, as $\sigma(a)$ is the denominator in the calculation of γ and $\sigma(a) \wedge \sigma(b) \geq \sigma(ab)$.

In order to effectively incorporate this pruning technique, the inferences R for v must be produced in order of descending antecedent length so that, if $\gamma(i \Rightarrow j) < minconf$, then all antecedents $i' \mid i' \subset i$ will not be considered. The process is more intuitive if considered from the other direction, that of generating the inferences in order of increasing consequent length, due to its nonmonotonicity. For example, given the derivation of a valid inference $\gamma(i \Rightarrow j) \geq minconf$ from v, then all other inferences derived from v of form $i' \Rightarrow j'$ must be valid where $j' \subset j$. This is true for the same case given that $j' \subset j$, then $i' \supset i$. Inference generation then becomes an iterative process incorporating a nonmonotonic constraint (in a similar manner to Apriori) where the inferences of consequent length κ are based on the valid inferences of consequent length, $\kappa - 1$, discovered during the previous iteration.

Table II. Summary of Condensed Representation and Incomplete Algorithms

Name	Author	Year	Base	Principle	Contribution
Condensed Representation Algorithms					
Close	Pasquier et al.	1999	Apriori	Closed Sets	Gauloise Connection
A-Close	Pasquier et al.	1999	Close	Closed Set	
CLOSET	Pei et al.	2000	FP-Growth	Closed Set	
PASCAL	Bastide et al.	2000	A priori	Counting Inference	
MinEx	Boulicant et al.	2001	A priori	δ-free sets	
HLinEx	Bykowski & Rigotti	2001	A priori	Disjunction free sets	
VLinEx	Bykowski & Rigotti	2001	Depth Project	Disjunction free sets	
CHARM	Zaki & Hsiao	2002	Eclat	Closed Set	
NDI	Calders & Goethals	2002	Apriori	Deduction rules	
CLOSET+	Wang et al.	2003	CLOSET	Closed Set	Hybrid tree projection, item skipping & efficient subset-checking structures
FPClose	Grahne & Zhu	2003	CLOSET	Closed Set	Optimised subset testing
Incomplete Set algorithms					
OCD	Mannila et al.	1994	OCD	Sampling	Naive extension
—	Toivonen	1996	Partition	Sampling	Negative border
All-MFS	Gunopulos et al.	1997	Depth Project	MFS	Random walks
MaxMiner	Bayardo	1998	Apriori	MFS	Upward Closure Principle
Pincer Search	Lin & Kedem	1998	Apriori	MFS	Bi-directional search
A-CPA	Lin & Dunham	1998	AS-CPA	sampling	Naive extension
Max-eclat	Zaki et al.	2000	Eclat	MFS	
Max-clique	Zaki et al.	2000	Clique	MFS	
MAFIA	Burdick et al.	2001	Depth Project	MFS	Accrual optimisation
GenMax	Gouda & Zaki	2001	Depth Project	MFS	Progressive focusing & diffsets
FAST	Chen et al.	2002	—	sampling	Two-phase sampling technique
FPMax	Grahne & Zhu	2003	FP-Growth	MFS	
FPMax*	Grahne & Zhu	2003	FPMax	FP-Growth	Progressive focus & Optimised subset tests

In addition to this general optimization of rule generation, the literature presents three classes of technique that result in a more interesting or reduced result set—post-analysis filtering, the inclusion of user beliefs, and condensed representations—the first two of which are outside the scope of this survey on association mining fundamentals and are only briefly discussed for completeness in the following. Of interest here are condensed representation optimizations to the fundamental rule derivation technique outlined previously.

Postanalysis filtering was first addressed by Toivonen et al. [1995] who proposed a simple post-analysis inferencing technique in which a rule $r : i \Rightarrow j$ is removed if another rule $r' : i' \Rightarrow j'$ exists, where $i' = i$, $j' \subset j$ and $Tidlist(r) = Tidlist(r')$. An inference basis clustering technique was also proposed where the distance between cluster bases is the difference between their associated tidsets, further facilitating user interpretation. This initial pruning technique was subsequently incorporated into the rule pruning list, formalized by Shah et al. [1999] (rule 1), that uses heuristics to prune redundant rules from the discovered rule set (presented in the following). Other techniques postanalysis techniques of significance have also been proposed [Srikant and Agrawal 1996; Liu et al. 1999].

(1) If two inferences $i \Rightarrow j$ and $i \wedge m \Rightarrow j$ of similar strength exist, then $i \wedge m \Rightarrow j$ is redundant.

(2) If two inferences $i \Rightarrow j$ and $m \Rightarrow j$ with similar strength exist, $m \Rightarrow j$ is redundant if $i \Rightarrow m$ is valid and $m \Rightarrow i$ is not.

(3) If two inferences $i \Rightarrow j$ and $m \Rightarrow j$ with similar strength exist, they are considered *weak* inferences if both $i \Rightarrow m$ and $m \Rightarrow i$ are valid. A weak inference is one which is valid in terms of the quality heuristic, however, due to the possible presence of alternative causes, their validity may be questionable.
(4) If two inferences $i \Rightarrow j$ and $i \Rightarrow m \wedge j$ exist, then $i \Rightarrow j$ is redundant.
(5) If two inferences $i \Rightarrow m$ and $i \Rightarrow j$ exist, and further $m \Rightarrow j$ exists, then $i \Rightarrow j$ is redundant.

Padmanabhan and Tuzhilin [1998] reduce inference generation to unexpected rules based on a set of user-defined beliefs and the definition of a new unexpected heuristic. Although incorporating the user, the resultant algorithm ZoomUR is significant because, contrary to previous techniques, the unexpectedness heuristic is incorporated within analysis, reducing the rule production, instead of being implemented as a postanalysis pruning and summarization technique. The unexpectedness heuristic proposed states that given single consequent sets, a rule $i \Rightarrow j$ is unexpected with respect to a belief $x \Rightarrow y$ if the following conditions hold:

(1) If j and y logically contradict each other $j \cup y = \emptyset$;
(2) $|i \cup j| \geq minsup$, the union of the belief and rule antecedents is significant with respect to a user specified threshold, such as support;
(3) $i, x \Rightarrow j$ holds with the same level of significance as $i \Rightarrow j$.

ZoomUR constrains itemset generation, using breadth-first candidate generation for each belief, to those containing x and the contradictions of y from which only contradictory rules are derived, resulting in belief refinement rules of the form $i, x \Rightarrow j$. A second stage of inference generation then considers each unexpected rule generated and tries to determine all the other more general rules that are also unexpected of the form $i, x' \Rightarrow j$ such that $x' \subset x$.

The number of inferences generated in any mining run is often too many to be effectively presented and interpreted by the user. Rule inferencing constrains rule generation to nonredundant rules where a rule is redundant if it conveys the same or less general information than another rule of the same usefulness and relevance [Pasquier et al. 1999a]. These inferencing mechanisms should ideally be lossless, sound, and informative to provide an accurate condensed representation of all association rules [Kryszkiewicz 2002]. These techniques fall into three classes, representative cover, closed-set cover, and basic association rules each of which is presented in the following sections.

—*lossless*, if all association rules can be derived from its representation.
—*sound*, if all derivable rules belong to the set of all association rules.
—*informative*, if the interestingness heuristics for each derivable rule can be calculated by the representation.

Furthermore, to avoid confusion between inferences and the task of inferencing, within this section an inference will be referred to as a rule which, although incorrect due to its definitive implication, will suffice.

6.1. Representative Cover

Padmanabhan and Tuzhilin [2000] extend their previous work relating to the discovery of unexpected rules, ZoomUR [Padmanabhan and Tuzhilin 1998], to directly generate a minimal set of unexpected rules, based on inference under the assumption of monotonicity. Given two unexpected rules $a, b, c \Rightarrow d$ and $a, b \Rightarrow d$, then the

monotonicity assumption states that, if one antecedent implies the other (abc implies ab), then $a, b \Rightarrow d$ will hold on the subset of d defined by abc, and hence $a, b, c \Rightarrow d$ is redundant. Thus S is the minimal rule set of R if the following three conditions hold:

(1) $S \subseteq R$,
(2) $\forall r_i \in R, \exists\, s_i \in S \mid s_i \models_m x_i$,
(3) $\forall s_1, s_2 \in S, s_1 \not\models_m s_2$,

where $\models_m$ represents a logical implication with respect to the monotonicity assumption.

The resultant algorithm, MinZoomUR, extends ZoomUR, considering both belief refinement (MinZoomin) and subsequent generalization (MinZoomOut) in the generation of minimal unexpected patterns. This is achieved through the inclusion of a set of *exclusion rules* during itemset generation (MinZoomin), and the dynamic generation of inferences as valid itemsets are discovered which consider further constraints according to generalization principles (MinZoomOut).

Kryszkiewicz [1998] proposed the concept of representative association rules which are the smallest set of rules that cover all association rules satisfying $minconf$. This inferencing technique, shown to be lossless and sound by the author, is based on a cover operator denoted CV, where given the rule $i \Rightarrow j, j \neq \emptyset$ as defined in Equation 12.

$$CV(i \Rightarrow j) = \{i \cup z \Rightarrow v \mid z, v \subseteq j \ \wedge\ z \cap v \neq \emptyset \ \wedge\ v \neq \emptyset\} \tag{12}$$

$$RR = \{r \in AR \mid \neg\exists r' \in AR \ \wedge\ r' \neq r \ \wedge\ e \in CV(r')\} \tag{13}$$

Each rule in $CV(i \Rightarrow j)$ contains a subset of $i \cup j$ elements, their antecedent is a superset of i, while their consequent is a nonempty subset of j. The authors show that each rule in the cover is a valid association rule as their interestingness metrics must be greater than or equal to those of the covering rule. Thus if the covering rule r is valid, so too is each rule in the cover $CV(r)$. Given this notion of a cover operator, the set of representative association rules, RR, is the smallest set of rules that covers all association rules by means of the cover operator. Hence no representative association rule can belong in the cover of another association rule, where the complete set of association rules are denoted AR as defined in Equation 13.

The authors subsequently designed the FastGenAllRepresentatives algorithm which, given the set of all frequent itemsets (V) and $minconf$, will generate RR based on two properties (presented in the following) derived from the properties of the cover operator. The first property ensures that all rules discovered do not exist in a cover of any other rule that is longer or of the same length. The first element of this property guarantees that the rule does not belong in the cover of a longer rule, while element two guarantees that the rule does not exist in a rule of the same length. The second property is used within the algorithm for process optimization; it ensures that, if an itemset has a valid proper superset with the same support, σ, then it is not representative and is eliminated since it must belong in the superset's cover.

Property 1. Given $\emptyset \neq i \subset z \subseteq I$ and a rule $i \Rightarrow j \mid j = z \setminus i \ \wedge\ z \in V \ \wedge\ \gamma(i \Rightarrow j) \geq minconf$. Then the rule belongs in RR if:

(1) $maxSup \leq minsup$ or $maxSup/\sigma(i) < minconf$, where
$maxSup = max(\{sup(z') \mid z \subset z' \subseteq I\} \cup \{0\})$,
(2) $\neg\exists X', \emptyset \neq X' \subset X$, such that $(X' \Rightarrow Z \setminus X') \in V$.

Property 2. Given $\emptyset \neq i \subset z \subset z' \subseteq I$, if $\sigma(z) = \sigma(z')$, then no rule $i \Rightarrow j \mid j = z \setminus i \wedge z \in V \ \wedge\ \gamma(i \Rightarrow j) \geq minconf$ belongs in RR.

A similar cover operator, informative cover, is proposed by Cristofor and Simovici [2002], where given $r = i \Rightarrow j$ and $r' = i' \Rightarrow j'$ such that $i' \cup j' \subseteq i \cup j$, then if $\sigma(i') \leq \sigma(i)$, r informatively covers r'. The difference between these covers is that the informative cover doesn't require that the antecedent of the resulting cover rule be included in the antecedent of the covered rule, and it uses information regarding the antecedent support in constructing the covering set.

6.2. Deductive Rules

Recently Goethals et al. [2005] proposed nonderivable association rules, an extension to their work on deductive rules using the mathematical inclusion-exclusion principle as discussed in Section 4.3. This technique calculates the upper and lower bounds of a rule's confidence through application of the inclusion-exclusion principle; if these bounds are the same, then the rule is exactly deducible and hence redundant. This principle is based on knowledge of the rules subrules, where given $r = i \Rightarrow j$ and $r' = i' \Rightarrow j'$, the latter is said to be more general if $i' \subseteq i$ and $j' \subseteq j$. Given the rule $i \Rightarrow j$, the support of $i \cup j$ is bounded and then the upper and lower bound are divided by $\sigma(i)$, resulting in an upper and lower bound for the confidence of $i \Rightarrow j$.

Furthermore, the authors propose generalization and specializations of this technique. A lossy strategy is proposed in which rules are redundant if the bounded confidence interval is narrow, resulting in close but not exact confidence derivation. This can further reduce the resultant rule set but from which not all rules are derivable. Specialization or restricted pruning is also considered in which the antecedent or consequent must be the same, and although leading to larger result sets, can lead to improved understandability.

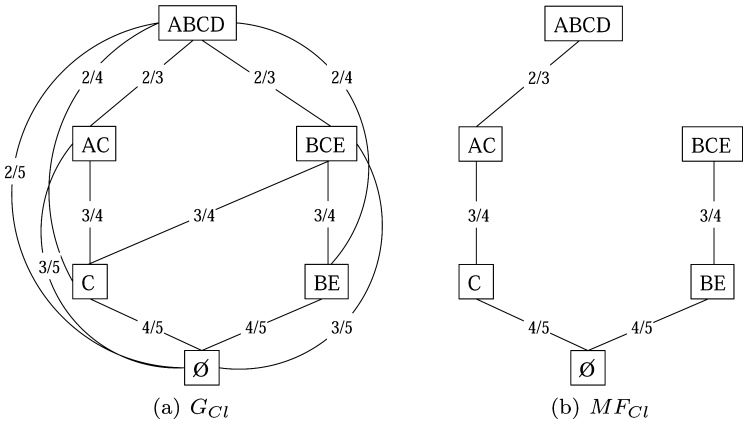
6.3. Closed Set Covers

Pasquier et al. [1999a] propose rule inferencing through the use of Galois closed sets to produce the smallest covers, or basis, for both exact (rule has $\gamma = 100\%$) and approximate rules, extending their previous work on closed itemset algorithms, see Section 4.1. Three equations are used to derive these basis, both requiring knowledge of the closed sets Cl within D. Exact rule inferencing is based on the Duguenne-Guigues Basis, while approximate rule inferencing is dealt with through the Proper Basis and the Structural Basis for approximate association rules.

Given a set of frequent pseudoclosed itemsets FP, Definition 1, the Duguenne-Guigues Basis (cover) for exact rules DG, is given in Equation (16). The resulting rules are nonredundant exact rules of minimal antecedent and maximal consequent, and the resultant cover is minimal, as there can be no complete set with fewer rules than there are frequent pseudoclosed itemsets [Ganter and Wille 1999]. The Proper Basis PB (Equation (17)) derives nonredundant approximate rules from Cl that exceed $minconf$. While the Structural Basis for approximate association rules is an independent abstraction of all valid approximate rules that can be used when the proper basis is large.

Definition 1 (*Frequent Pseudoclosed Itemsets*). An itemset, $i \in I$, is pseudo-closed *if and only if* $cl(i) \neq i$ and all pseudoclosed subsets of i, i', have a closure which a subset of i, $cl(i') \subset i$.

$$FP = \{i \subseteq I \mid \sigma(i) \geq minsup \ \wedge \ i \neq cl(i) \ \wedge \ \forall i' \in FP \text{ such that } i' \subset i \text{ then } cl(i') \subset i\}.$$



(a) G_{Cl} (b) MF_{Cl}

Fig. 20. Undirected graph (G_{Cl}) and maximal confidence spanning tree (MF_{Cl}).

$$V = \{i \subseteq I \mid i \in Cl\} \tag{14}$$

$$E = \{(i, j) \in V \times V \mid i \subset j \ \wedge \ \sigma(j)/\sigma(i) \geq minconf\} \tag{15}$$

$$DG = \{i \Rightarrow cl(i) \setminus i \mid i \in FP \ \wedge \ i \neq \emptyset\} \tag{16}$$

$$PB = \{r : i \Rightarrow j \setminus i \mid i, j \in Cl \ \wedge \ i \neq \emptyset \ \wedge \ i \subset j \ \wedge \ \gamma(r) \geq minconf\} \tag{17}$$

$$SB = \{r : i \Rightarrow j \setminus i \mid i, j \in V \ \wedge \ i \neq \emptyset \ \wedge \ i \subset j \ \wedge \ (i, j) \in E'\} \tag{18}$$

The Structural Basis for approximate association rules is derived from the maximal confidence spanning forest, MF_{Cl}, associated with Cl. Given an undirected graph $G_{Cl} = (V, E)$ where vertices V and edges E are defined in Equation (14) and 15, $MF_{Cl} = (V, E')$ is obtained by removing transitive edges and cycles, such that the edge remaining has maximal confidence, illustrated in Figure 20, derived from Pasquier et al. [1999a]. The structural basis, derived from MF_{Cl}, is then the set of association rules represented by the edges in MF_{Cl} as defined in Equation (20).

Although producing the minimal basis with respect to the number of extracted association rules, Bastide et al. [2000] extended this research proposing two new basis that discover rules of minimal antecedent and maximal consequent which they claim is of more interest. The new basis, the Generic Basis for exact rules and the Informative Basis for approximate association rules, are constructed from Cl and their generators, as shown in Equations (19) and (20), where an itemset i is a minimal generator, Gen, of a closed itemset j *iff* $Cl(i) = j$, and there exists no subset i' of i such that $Cl(i') = j$. Furthermore, the authors propose a transitive reduction of IB, denoted RI, in which only closures of i that are immediate predecessors of j, denoted by $\lhd$, are valid (Equation (21)).

$$GB = \{i \Rightarrow j \setminus i \mid j \in Cl \ \wedge \ i \in Gen_j \ \wedge \ i \neq j\} \tag{19}$$

$$IB = \{i \Rightarrow j \setminus i \mid j \in Cl \ \wedge \ i \in Gen_{Cl} \ \wedge \ cl(i) \subset j\} \tag{20}$$

$$RI = \{i \Rightarrow j \setminus i \mid j \in Cl \ \wedge \ i \in Gen_{Cl} \ \wedge \ cl(i) \lhd j\} \tag{21}$$

Zaki [2004] proposes a variation to this in which a redundant rule is one for which a more general rule exists with the same support and confidence, where a rule r is more general than r', if r' can be generated from r by appending items to either the antecedent or consequent. The inferencing technique proposed uses closed sets Cl and

their minimal generators, *Gen*, from which the set of potential exact (*PE*) and approximate (*PA*) rules are derived. Given two closed sets $i, j \mid i \subset j$ and *Gen*, denoted G, these equations are provided in Equation (22) and (23). By finding the most general rules from this set (with the same support and confidence), it is argued that the set of nonredundant rules between the pair of closed itemsets has been found.

$$PE = \{b \Rightarrow d \setminus b \mid b \in G_j \ \wedge \ d \in G_i \ \wedge \ b \cap d \neq \emptyset \ \wedge \ cl_{d \setminus b} = i \ \wedge \ cl_{d \cup b} = j \quad (22)$$

$$PA = \{b \Rightarrow d \setminus b \mid b \in G_i \ \wedge \ d \in G_j \ \wedge \ b \cap d \neq \emptyset \ \wedge \ cl_b = i \ \wedge \ cl_{d \cup b} = j \quad (23)$$

6.4. Basic Association Rules

Li and Hamilton [2004] suggest that current inferencing techniques are suboptimal as they do not consider Armstrong's rules of inference for functional dependencies [Armstrong 1974]. The authors subsequently propose that the minimal set of association rules are analogous to minimal functional dependancies and propose, a set of inference rules based on restricted conditional probability distribution that address Armstrong's axioms. Previously, Bastide [2000] argued that these axioms were inapplicable as they did not take interestingness metrics into account, however, the proposed technique addresses this by enabling the inference of the confidence of rules.

The GenBR algorithm results in the generation of Basic Association Rules which are nonredundant single consequent or canonical rules ($i \Rightarrow j$), where i is conditionally minimal with respect to j, that is, $\nexists i' \subset i$, such that:

(1) $\gamma(i \Rightarrow j) = \gamma(i' \Rightarrow j)$,
(2) $\forall m \mid m \subseteq i - i', \gamma(i \Rightarrow j) = \gamma(i', m \Rightarrow j)$.

Through the subsequent derivation of Basic Association Rule Classes BC, where BC_i^r represents the set of items in each BC^r, the authors propose that all association rules and their associated confidence can be derived using the following set of rules of inference for functional dependencies, where p and q signify confidence. Note that the *contraction* inference relates not to basic association rule classes but FastGenRules [Agrawal and Srikant 1994] (see Section 6) denoted by AR, which the authors include for completeness.

Inference Rule	Premise	Conclusion
Augmentation	$i \stackrel{p}{\Longrightarrow} j \in BC^r \ \wedge \ \forall i' \subseteq BC_i^r \setminus \{i, j\}$	$ii' \stackrel{p}{\Longrightarrow} j$
Pseudo-Transitivity	$i \stackrel{p}{\Longrightarrow} j \in BC^r \ \wedge \ i, n \stackrel{q}{\Longrightarrow} m \in BC^r$	$i, n \stackrel{q}{\Longrightarrow} m$
Additivity	$i \stackrel{p}{\Longrightarrow} j \in BC^r \ \wedge \ n \stackrel{q}{\Longrightarrow} m \in BC^r$	$i, n \stackrel{pq}{\Longrightarrow} j, m$
Contraction	$i \stackrel{p}{\Longrightarrow} j \in AR \ \wedge \ i, j \stackrel{q}{\Longrightarrow} m \in AR$	$i \stackrel{pq}{\Longrightarrow} j, m$
Right Union	$i \stackrel{p}{\Longrightarrow} j \in BC^r \ \wedge \ i \stackrel{q}{\Longrightarrow} m \in BC^r$	$i \stackrel{pq}{\Longrightarrow} j, m$
Left Union	$i \stackrel{p}{\Longrightarrow} m \in BC^r \ \wedge \ j \stackrel{q}{\Longrightarrow} m \in BC^r$	$i, j \stackrel{pq}{\Longrightarrow} m$

Table III summarizes the inference generation algorithms discussed in this section.

7. SUMMARY

A decade on from the seminal work of Agrawal et al. [1993, 1994], association mining has become a mature field of research with diverse branches of specialization. The fundamentals of association mining are now well established and, with some important exceptions. there appears little current research involving the improvement of general itemset identification or rule generation.

Table III. Summary of Inference Generation Algorithms

Name	Author	Year	Base	Contribution
FastGenRules	Agrawal et al.	1994	Constrained generation	Optimised generation of all valid inferences
	Toivonen et al.	1995	Post-Analysis Filtering	removal of redundant rules
ZoomUR	Padmanabhan et al.	1998	Domain Knowledge	Belief inclusion in inference generation
GenAllRepresentatives	Kryszkiewicz	1998	Representative Cover	Representative association rules
	Pasquier et al.	1999	Closed-set cover	Duguenne-Guigues Basis, Proper Basis, Structual Basis
	Shah et al.	1999	Post-Analysis Filtering	removal of redundant rules
	Bastide et al.	2000	Closed-set cover	Generic Basis, Informative Basis
MinZoomUR	Padmanabhan et al.	2000	Representative Cover	Minimal set of unexpected rules
CoverRules	Cristofor et al.	2002	Representative Cover	Informative Cover
	Zaki	2004	Close-set cover	Minimal closed rules
GenBR	Li et al.	2004	Basic Association Rules	Functional dependancy inferencing
	Goethals et al.	2005	Deductive rules	Non-derivable association rules

This survey has provided an organization of the significant fundamental contributions made within association mining research over this time. It highlights the maturity of the fundamental principles within itemset identification and rule generation.

Current research appears to focus on the specialization of fundamental association mining algorithms, many areas of which are still emerging. These include fields such as, measures of interest, the inclusion of domain knowledge and semantics, quantitative mining, disassociation mining, privacy mining, incremental mining, iterative and interactive or guided mining, and higher order mining (the mining of rulesets).

REFERENCES

AGRAWAL, R., IMIELINSKI, T., AND SWAMI, A. 1993. Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD International Conference on the Management of Data*. Washington DC, P. Buneman and S. Jajodia, Eds. ACM Press, 207–216.

AGRAWAL, R. AND SRIKANT, R. 1994. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB'94)*, Santiago, Chile. B. Bocca, M. Jarke, and C. Zaniolo, Eds. Morgan Kaufmann, 487–499.

AGRAWAL, R. C., AGGARWAL, C. C., AND PRASAD, V. V. V. 1999. A tree projection algorithm for generation of frequent itemsets. *High Performance Data Mining Workshop*. Puerto Rico. ACM Press.

AGRAWAL, R. C., AGGARWAL, C. C., AND PRASAD, V. V. V. 2000. Depth first generation of long patterns. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Boston, MA. ACM Press, 108–118.

ARMSTRONG, W. W. 1974. Dependency structures of data base relationships. In *Proceedings of the 6th International Federation for Information Processing Congress (IFIP)*. Vol. 74. North Holland, Amsterdam, The Netherlands, 580–583.

BASTIDE, Y., PASQUIER, N., TAOUIL, R., STUMME, G., AND LAKHAL, L. 2000. Mining minimal non-redundant association rules using frequent closed itemsets. In *Proceedings of the 1st International Conference on Computational Logic, (CL'00)*, J. W. Lloyd, V. Dahl, U. Furbach, M. Kerber, K.-K. Lau, C. Palamidessi,

L. M. Pereira, Y. Sagiv, and P. J. Stuckey, Eds. Lecture Notes in Computer Science, vol. 1861. Springer, Berline, Germany, 972–986.

BASTIDE, Y., TAOUIL, R., PASQUIER, N., STUMME, G., AND LAKHAL, L. 2000. Mining frequent patterns with counting inference. *SIGKDD Explorations 2*, 2, 66–75.

BAYARDO JR, R. 1998. Efficiently mining long patterns from databases. *ACM SIGMOD International Conference on the Management of Data (SIGMOD'98)*. Seattle, WA, ACM Press, 85–93.

BAYARDO JR, R. AND AGRAWAL, R. 1999. Mining the most interesting rules. In *Proceedings of the 5th International Conference on Knowledge Discovery and Data Mining* San Diego, CA, S. Chaudhuri and D. Madigan, Eds. ACM Press, 145–154.

BODON, F. 2003. A fast Apriori implementation. In *Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI'03)*, Melbourne, Fl. B. Goethals and M. J. Zaki, Eds. CEUR Workshop Proceedings, vol. 90. IEEE Press.

BOULICANT, J.-F., BYKOWSKI, A., AND RIGOTTI, C. 2001. Free-sets: A condensed representation of boolean data for the approximation of frequency queries. *Data Mining Know. Discov. 7*, 1, 5–22.

BRIN, S., MOTWANI, R., ULLMAN, J. D., AND TSUR, S. 1997. Dynamic itemset counting and implication rules for market basket data. *SIGMOD Record 26*, 2, 255–276.

BURDICK, D., CALIMLIM, M., AND GEHRKE, J. 2001. Mafia: A maximal frequent itemset algorithm for transactional databases. In *Proceedings of the 17th International Conference on Data Engineering*, Heidelburg, Germany, IEEE Press, 443–452.

BYKOWSKI, A. AND RIGOTTI, C. 2001. A condensed representation to find frequent patterns. In *Proceedings of the 20th ACM SIGMOD-SIGACT-SIGART Symposium on the Principles of Database Systems*, Santa Barbara, CA. ACM Press, 267–273.

CALDERS, T. AND GOETHALS, B. 2002. Mining all non-derivable frequent itemsets. In *Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery*, Helsinki, Finland. T. Elomaa, H. Mannila, and H. Toivonen, Eds. Lecture Note in Artificial Intelligence, vol. 2431. Springer, Berlin, Germany, 74–85.

CHEN, B., HAAS, P., AND SCHEUERMANN, P. 2002. A new two-phase sampling based algorithm for discovering association rules. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Edmonton, Alberta, Canada, ACM Press, 462–468.

CHOUDHURI, S., DATAR, M., MOTWANI, R., AND NARASAYYA, V. 2001. Overcoming limitations of sampling for aggregation queries. In *Proceedings of the 17th International Conference on Data Engineering (ICDE)*, Heidelburg, Germany. IEEE Press, 534–542.

CRISTOFOR, L. AND SIMOVICI, D. 2002. Generating an informative cover for association rules. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM'02)*, Washington, DC. IEEE Computer Society, 597–613.

DESHASPE, L. AND TOIVONEN, H. 1998. Frequent query discovery: A unifying approach to association rule mining. Tech. rep. CW-258, Department of Computer Science, Katholieke Universiteit Leuven.

DONG, G. AND LI, J. 1998. Interestingness of discovered association rules in terms of neighbourhood-based unexpectedness. In *Proceedings of the 2nd Pacific-Asia Conference on Research and Development in Knowledge Discovery and Data Mining(PAKDD'98)*, Melbourne, Australia, X. Wu, R. Kotagiri, and K. Korb, Eds. Lecture Notes in Artificial Intelligence, vol. 1394. Springer, Berline, Germany, 72–86.

DUNKEL, B. AND SOPARKAR, N. 1999. Data organization and access for efficient data mining. In *Proceedings of the 15th International Conference on Data Engineering*, Sydney, Australia. IEEE, 522–532.

FREITAS, A. 1999. On rule interestingness measures. *Knowl. Based Syst. 12*, 5-6, 309–315.

GANTER, G. AND WILLE, R. 1999. *Formal Concept Analysis: Mathematical Foundations*. Springer, Berlin, Germaney.

GARDARIN, G., PUCHERAL, P., AND WU, F. 1998. Bitmap based algorithms for mining association rules. *14th Bases de Donnes Avances (BDA'98)*. Hammamet, Tunisia, Springer, Berlin, Germany, 157–176.

GIBBONS, P. 2001. Distinct sampling for highly-accurate answers to distinct values queries and event reports. In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB'01)*, Rome, Italy. Morgan Kaufmann, 541–550.

GOETHALS, B., MUHONEN, J., AND TOIVONEN, H. 2005. Mining non-derivable association rules. *5th SIAM International Conference on Data Mining (SDM'05)*. Newport Beach, CA.

GOPALAN, R. P. AND SUCAHYO, Y. G. 2002. ITL-mine: Mining frequent itemsets more efficiently. In *Proceedings of the 2002 Internatiional Conference on Fuzzy Systems and Knowledge Discovery*, Singapore, L. Wang, S. Halgamuge, and X. Yao, Eds. Vol. 1. Springer, Berlin, Germany, 167–172.

GOUDA, K. AND ZAKI, M. J. 2001. Efficiently mining maximal frequent itemsets. In *Proceedings of the IEEE International Conference on Data Mining*, San Jose. IEEE Press, CA.

GOULBOURNE, G., COENEN, F., AND LENG, P. 2000. Algorithms for computing association rules using a partial support tree. *Knowl. Based Syst. 13*, 2–3, 141–149.

GRAHNE, G. AND ZHU, J. 2003a. Efficiently using prefix-trees in mining frequent itemsets. In *Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI'03)*, Melbourne, FL. B. Goethals and M. J. Zaki, Eds. Vol. 90. IEEE Press.

GRAHNE, G. AND ZHU, J. 2003b. High performance mining of maximal frequent itemsets. *6th SIAM International Workshop on High Performance Data Mining (HPDM'03)*. San Francisco, CA.

GUNOPULOS, D., MANNILA, H., AND SALUJA, S. 1997. Discovering all most specific sentences by randomised algorithms extended abstract. In *Proceedings of the 6th International Conference on Database Theory*. Delphi, Greece, F. Afrati and P. Kolaitis, Eds. Springer, Berlin, Germany. 251–229.

HAN, J. AND PEI, J. 2000. Mining frequent patterns by pattern growth: Methodology and implications. *SIGKDD Explorations 2*, 2, 14–20.

HILDERMAN, R. J. AND HAMILTON, H. J. 1999. Heuristic measures of interestingness. In *Proceedings of the 3rd European Conference on Principles of Knowledge Discovery in Databases (PKDD'99)*, J. Zytkow and J. Rauch, Eds. Lecture Notes in Artificial Intelligence, vol. 1704. Springer, Berlin, Germany. 232–241.

HILDERMAN, R. J. AND HAMILTON, H. J. 2001. Evaluation of interestingness measures for ranking discovered knowledge. In *Proceedings of the 5th Pacific-Asia Conference on Knowledge Discovery and Data Mining, (PAKDD'01)*, Hong Kong, China, D. W.-L. Cheung, G. J. Williams, and Q. Li, Eds. Lecture Notes in Computer Science, vol. 2035. Springer, Berlin, Germany, 247–259.

HIPP, J., GÜNTZER, U., AND NAKHAEIZADEH, G. 2000a. Algorithms for association rule mining—a general survey and comparison. *SIGKDD Explorations 2*, 1, 58–65.

HIPP, J., GÜNTZER, U., AND NAKHAEIZADEH, G. 2000b. Mining association rules: Deriving a superior algorithm by analysing today's approaches. In *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD'00)*, Lyon, France. D. Zighed, J. Komorowski, and J. Zytkow, Eds. LNAI, vol. 1910. Springer, Berlin, Germany, 159–168.

HOUTSMA, M. AND SWAMI, A. 1993. Set oriented mining of association rules. Tech. rep. RJ 9567, IBM Almaden Research Centre.

KIVINEN, M. AND MANNILA, H. 1994. The power of sampling in knowledge discovery. In *Proceedings of the 13th ACM SIGACT-SIGMOD-SIGART Symposium on the Principles of Database Systems (PODS'94)*. Minneapolis, NM, ACM Press, 77–85.

KOSTERS, W. A. AND PIJLS, W. 2003. Apriori, a depth first implementation. *IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI'03)*. Melbourne, FL, B. Goethals and M. J. Zaki, Eds. IEEE Press.

KRYSZKIEWICZ, M. 1998. Fast discovery of representative association rules. In *Proceedings of the Rough Sets and Current Trends in Computing, 1st International Conference, (RSCTC'98)*, Warsaw, Poland, L. Polkowski and A. Skowron, Eds. Lecture Notes in Computer Science, vol. 1424. Springer, Berlin, Germany, 214–221.

KRYSZKIEWICZ, M. 2002. Concise representations of association rules. *ESF Exploratory Workshop on Pattern Detection and Discovery*. London, UK, Springer, Berlin, Germany. 92–109.

LI, G. AND HAMILTON, H. J. 2004. Basic association rules. In *Proceedings of the 4th SIAM International Conference on Data Mining (SDM'04)*. Orlando, FL.

LIN, D.-I. AND KEDEM, Z. 1998. Pincer search: A new algorithm for discovering the maximum frequent set. In *Proceedings of the 6th International Conference on Extending Database Technology, (EDBT'98)*. Valencia, Spain, 385–392.

LIN, J. L. AND DUNHAM, M. H. 1998. Mining association rules: Anti skew algorithms. In *Proceedings of the 14th International Conference on Data Engineering*. Orlando, FL, IEEE Computer Society Press, 486–493.

LIU, B., HSU, W., AND MA, Y. 1999. Pruning and summarizing the discovered rules. In *Proceedings of the 5th ACM SIGMOD Conference on Data Mining and Knowledge Discovery*.

LIU, J., PAN, Y., WANG, K., AND HAN, J. 2002. Mining frequent item sets by opportunistic projection. In *Proceedings of the Knowledge Discovery in Databases*. Edmonton, Canada, Vol. 31. ACM Press, 97–102.

LUCCHESE, C., ORLANDO, S., PALMERINI, P., PEREGO, R., AND SILVESTRI, F. 2003. kDCI: A multi-strategy algorithm for mining frequent sets. In *Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI'03)*, Melbourne, FL B. Goethals and M. J. Zaki, Eds. Vol. 90. IEEE Press.

MANNILA, H., TOIVONEN, H., AND VERKAMO, A, I. 1994. Efficient algorithms for discovering association rules. In *Proceedings of the AAAI Workshop on Knowledge Discovery in Databases*, Seattle, WA, M. Fayyad, U and R. Uthurusamy, Eds, 181–192.

MUELLER, A. 1995. Fast sequential and parallel algorithms for association rule mining: a comparison. Tech. rep. CS-TR-3515, Department of Computer Science, University of MaryLand, College Park, MD.

ORLANDO, S., PALMERINI, P., AND PEREGO, R. 2001a. DCI: A hybrid algorithm for frequent itemset counting. Tech. rep. CS-01-9-2001, Dipartimento. di Informatica, Universita Ca Foscari.

ORLANDO, S., PALMERINI, P., AND PEREGO, R. 2001b. Enhancing the Apriori algorithm for frequent set counting. In *Proceedings of the International Conference on Data Warehousing and Knowledge Discovery*, Munich, Germany, Y. Kambayashi, W. Winiwarter, and M. Arikawa, Eds. Springer, Berlin, Germany, 71–82.

OZEL, S. A. AND GUVENIR, H. A. 2001. An algorithm for mining association rules using perfect hashing and database pruning. In *10th Turkish Symposium on Artificial Intelligence and Neural Networks*, Gazimagusa, T.R.N.C., A. Acan, I. Aybay, and M. Salamah, Eds. Springer, Berlin, Germany, 257–264.

PADMANABHAN, B. AND TUZHILIN, A. 1998. A belief driven method for discovering unexpected patterns. In *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining (KDD-98)* New York, NY. AAAI Press, 94–100.

PADMANABHAN, B. AND TUZHILIN, A. 2000. Small is beautiful: Discovering the minimal set of unexpected patterns. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Boston, MA. R. Ramakrishnan, S. Stolfo, R. Bayardo, and I. Parsa, Eds. ACM Press, 54–63.

PARK, J. S., CHEN, M. S., AND YU, P. S. 1997. Using a hash-based method with transaction trimming and database scan reduction for mining association rules. *IEEE Trans. Knowl. Data Eng. 9*, 5, 813–825.

PASQUIER, N., BASTIDE, Y., TAOUIL, R., AND LAKHAL, L. 1999a. Closed set based discovery of small covers for association rules. In *Proceedings of the 15th Conference on Advanced Databases*. Springer. Bordeaux, France, 361–381.

PASQUIER, N., BASTIDE, Y., TAOUIL, R., AND LAKHAL, L. 1999b. Discovering frequent closed itemsets for association rules. In *Proceedings of the 7th International Conference on Database Theory (ICDT'99)*. Jerusalem, Israel, Springer, Berlin, Germany, 398–416.

PASQUIER, N., BASTIDE, Y., TAOUIL, R., AND LAKHAL, L. 1999c. Efficient mining of association rules using closed itemset lattices. *Informa. Syst. 24*, 1, 25–46.

PEI, J., HAN, J., AND LAKSHMANAN, L. V. S. 2001. Mining frequent itemsets with convertible constraints. In *Proceedings of the 17th International Conference on Data Engineering (ICDE'01)*. Heidelberg, Germany. IEEE Computer Society Press, 433–442.

PEI, J., MAO, R., HU, K., AND ZHU, H. 2000. Towards data mining benchmarking: a test bed for performance study of frequent pattern mining. In *ACM SIGMOD International Conference on the Management of Data (SIGMOD'00)*, Dallas, TX. W. Chen, J. Naughton, and P. A. Bernstein, Eds. ACM Press, 592.

PIETRACAPRINA, A. AND ZANDOLIN, D. 2003. Mining frequent itemsets using patricia tries. In *Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI'03)*, Melbourne, FL, B. Goethals and M. J. Zaki, Eds.

PIJLS, W. AND BIOCH, J. C. 1999. Mining frequent itemsets in memory resident databases. In *11th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC'99)*, Kasteel Vaeshartelt, Maastricht, The Netherlands. E. Postma and M. Gyssens, Eds. Springer, Berlin, Germany, 75–82.

RODDICK, J. F. AND RICE, S. P. 2001. What's interesting about cricket?—on thresholds and anticipation in discovered rules. *SIGKDD Explorations 3*, 1, 1–5.

RYMON, R. 1992. Search through systematic set enumeration. In *Proceedings of the 3rd International Conference on the Principles of Knowledge Representation and Reasoning*. Cambridge, MA. Morgan Kaufmann 539–550.

SAHAR, S. 1999. Interestingness via what is not interesting. In *Proceedings of the 5th International Conference on Knowledge Discovery and Data Mining*, San Diego, CA. S. Chaudhuri and D. Madigan, Eds. ACM Press, 332–336.

SAVASERE, A., OMIECINSKI, E., AND NAVATHE, S. 1995. An efficient algorithm for mining association rules in large databases. In *Proceedings of the 21st International Conference on Very Large Data Bases, (VLDB'95)*, Zurich, Switzerland. U. Dayal, P. M. D. Gray, and S. Nishio, Eds. Morgan Kaufmann, 432–444.

SHAH, D., LAKSHMANAN, L. V. S., RAMAMRITHAM, K., AND SUDARSHAN, S. 1999. Interestingness and pruning of mined patterns. In *Proceedings of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, Philadelphia, PA. K. Shim and R. Srikant, Eds. ACM Press.

SHENOY, P., HARITSA, J. R., SUDARSHAN, S., BHALOTIA, G., BAWA, M., AND SHAH, D. 2000. Turbo-charging vertical mining of large databases. In *Proceedings of the 2000 ACM SIGKDD International Conference on Management of Data* Dallas TX. W. Chen, J. Naughton, and P. A. Bernstein, Eds. Vol. 29. ACM Press, 22–33.

SILBERSCHATZ, A. AND TUZHILIN, A. 1995. On subjective measures of interestingness in knowledge discovery. In *Proceedings of the 1st International Conference on Knowledge Discovery and Data Mining (KDD-95)*, Montreal, Quebec, Canada. U. M. Fayyad and R. Uthurusamy, Eds. AAAI Press, Menlo Park, CA, 275–281.

SILBERSCHATZ, A. AND TUZHILIN, A. 1996. What makes patterns interesting in knowledge discovery systems? *IEEE Trans. Knowl. Data Eng. 8*, 6, 970–974.

SRIKANT, R. AND AGRAWAL, R. 1996. Mining quantitative association rules in large relational tables. In *Proceedings of the ACM SIGMOD International Conference on the Management of Data*, Montreal, Canada. H. V. Jagadish and B. Mumick, Eds. 1–12.

SUCAHYO, Y. G. AND GOPALAN, R. P. 2003. CT-ITL: Efficient frequent item set mining using a compressed prefix tree with pattern growth. In *Proceedings of the 14th Australasian Database Conference*. K. Dieter-Schewe and X. Zhou, Eds. CRPIT, vol. 25. Australian Computer Society Inc., Adelaide, Australia, 95–105.

TOIVONEN, H. 1996. Sampling large databases for association rules. In *22nd International Conference on Very Large Data Bases, (VLDB'96)*, Mumbai (Bombay), India. T. Vijayaraman, P. Buchmann, C. Mohan, and N. Sarda, Eds. Morgan Kaufmann, 134–141.

TOIVONEN, H., KLEMETTINEN, M., RONKAINEN, P., HATONEN, K., AND MANNILA, H. 1995. Pruning and grouping of discovered association rules. *ECML'95 Workshop on Statistics, Machine Learning, and Knowledge Discovery in Databases*. Heraklion, Greece, 47–52.

WANG, J., HAN, J., AND PEI, J. 2003. CLOSET+: Searching for the best strategies for mining frequent closed itemsets. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Washington, DC. L. Getoor, T. E. Senator, P. Domingos, and C. Faloutsos, Eds. ACM Press, 236–245.

WANG, K., TANG, L., HAN, J., AND LIU, J. 2002. Top down FP-growth for association rule mining. In *Proceedings of the 6th Pacific Asia Conference on Knowledge Discovery and Data Mining, (PAKDD'02)*, Taipei, Taiwan. B. L. Ming-Shan Cheng, Philip S. Yu, Eds. Vol. 2336. Springer, Berlin, Germany, 334–340.

YEN, S. J. AND CHEN, A. L. P. 1996. An efficient approach to knowledge discovery from large databases. In *Proceedings of the IEEE/ACM International Conference on Parallel and Distributed Information Systems*. ACM Press, 8–18.

ZAÏANE, O. R. AND EL-HAJJ, M. 2003. Cofi-tree mining: A new approach to pattern growth with reduced candidacy generation. In *Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI'03)*, Melbourne, FL. B. Goethals and M. J. Zaki, Eds.

ZAKI, M. J. 1999. Parallel and distributed association mining: A survey. *IEEE Concurrency* (Special Issue on Parallel Mechanisms for Data Mining.) 7, 4, 14–25.

ZAKI, M. J. 2000. Scalable algorithms for association mining. *IEEE Trans. Knowl. Data Eng. 12*, 3, 372–390.

ZAKI, M. J. 2001. SPADE: An efficient algorithm for mining frequent sequences. *Machine Learn. 42*, 1/2, 31–60.

ZAKI, M. J. 2004. Mining non-redundant association rules. *Data Mining Knowl. Disco. 9*, 3, 223–248.

ZAKI, M. J. AND HSIAO, C.-J. 2002. CHARM: An efficient algorithm for closed itemset mining. In *Proceedings of the 2nd SIAM International Conference on Data Mining (SDM'02)*, Arlington, VA. R. L. Grossman, J. Han, V. Kumar, H. Mannila, and R. Motwani, Eds. SIAM, 457–473.

ZAKI, M. J., PARTHASARATHY, S., OGIHARA, M., AND LI, W. 1997. New algorithms for fast discovery of association rules. In *3rd International Conference on Knowledge Discovery and Data Mining (KDD'97)*. Newport Beach, CA. AAAI Press, 283–286.

Received July 2004; revised August 2005; accepted February 2006