CrystalBall: A Framework for Mining Variants of Association Rules[*]

Kok-Leong Ong **Wee-Keong Ng** **Ee-Peng Lim**

School of Computer Engineering, Nanyang Technological University,
Nanyang Avenue, N4-B3C-14, Singapore 639798
Email: ongkl@pmail.ntu.edu.sg

Abstract

The mining of informative rules calls for methods that include different attributes (e.g., weights, quantities, multiple-concepts) suitable for the context of the problem to be analyzed. Previous studies have focused on algorithms that considered individual attributes but ignored the information gain in each rule when the interaction of two or more attributes are taken into account. Motivated by the above, we developed a framework called CrystalBall that supports declarative mining of different rules (i.e., variants) involving several attributes. It eliminates the time and cost of engineering algorithms as practiced in previous studies, and introduces a foundation for cross-variant enhancements. The framework consists of a generic rule mining engine (VI), and a variant description language (VDL) for defining attribute-specific behavior. Besides demonstrating the flexibility of the framework, we also discuss the experimental studies, the limitations of the framework, as well as future work in the paper.

Keywords: Framework, Apriori, Association Rules

1 Introduction

The mining of association rules has been a well researched problem that can be classified broadly into two categories. The first category focuses primarily on mining different variants of rules (i.e., rules that consider different attributes such as weights, quantities, multiple-concepts) and how to efficiently discover them (Agrawal & Srikant 1994, Cai, Fu, Cheng & Kwong 1998, Han & Fu 1995, Han, Pei & Yin 2000, Mannila, Toivonen & Verkamo 1995, Park, Chen & Yu 1995, Savasere, Omiecinski & Navathe 1995, Zaiane, Han & Zhu 2000). Research in the latter category explores other aspects of rule mining such as specifying constraints (Ng, Lakshmanan & Han 1998, Ng, Lakshmanan & Han 1999, Pei & Han 2000) and evaluating its interestingness (Dong & Li 1998, Liu, Hsu & Ma 1999, Liu, Hu & Hsu 2000). Generally, studies in both categories focus on a single variant at a time, and include specific techniques to find the rules or evaluate the interestingness efficiently. In many data mining tasks, domain experts are often interested in getting the most informative rule. This may be achieved by considering different attributes at the same time. However, doing so may lead to the loss of efficient discovery mechanisms and engages the domain analyst in the time and resource-consuming cycle of engineering an algorithm. Hence, previous work has ignored the fundamental need for different variants to interact with one another to achieve the ultimate goal of knowledge discovery. In the following examples, we illustrate scenarios extracted from real world applications that call for the need to mine informative rules by considering multiple attributes simultaneously.

Example 1 (Mining Supermarket Data with Recurrent Weighted Items) Suppose an analyst is interested in items that have been co-purchased with a certain promotional item. For a more accurate analysis, items are assigned weights to improve the discovery of frequent itemsets that represent co-purchases. In additional to knowing what is bundled, he is also interested in how many of each item have been purchased together. These two attributes (i.e., weights and quantity) have never been considered together by any algorithms despite the informative rule it produces.

Example 2 (Finding Abstract Spatial Objects with Multiple Minimum Support) Suppose an analyst is interested to find the co-occurrence of a class of spatial objects with respect to another. Since some classes of spatial objects appear frequently while others occur rarely, using a single support threshold is inappropriate. Therefore, the analyst would like to specify individual support for each class of spatial objects. To find such rules requires the consideration of two attributes — multiple concept levels and individual support. Existing algorithms do not find such rules.

The key problem faced by domain experts in the above scenarios is that they have to either settle for what is in the data mining package, or expend time and resources to engineer the appropriate algorithm. Neither is desirable and the problem resurfaces whenever new variants are needed or discovered. Another important point made by the examples is the need to consider interactions among different attributes associated with the data. Current methodologies have ignored this issue and created several ad-hoc algorithms in which most data mining applications support only a subset. Hence, a proliferation of algorithms for different variants becomes daunting and a framework for unifying and eliminating the process, we believe, would be a welcome development. This is what CrystalBall is about – a framework to discover informative rules without the cost of engineering algorithms. This would be the solution that domain experts desire. Our contributions, set in the above context, answer the following questions:

[*] This work was supported by SingAREN, Project M48020004.

- **How do we eliminate the daunting process of developing algorithms, and at the same time, continue to discover informative rules?** We propose a rule mining engine (called VI) that is generic in nature. This engine does not make any assumption about the data nor the attributes under consideration. Its behavior is entirely dependent on the domain expert's constraints. The algorithm, which is an extension of the Apriori (Agrawal & Srikant 1994), is given in Figure 1 and discussed in Section 2.
- **How is the domain knowledge conveyed to the data mining engine?** The analyst, with domain knowledge about the data and the associated attributes, considers how each attribute influences the results of the data mining task. This is done declaratively via the Variant Description Language (or VDL in short) that is discussed in Section 2.2 and Section 3.
- **Can the framework be realized in practice?** We present an implementation as a proof of concept on the framework's feasibility. We also conduct performance testing to determine the overheads involved as a result of abstractions about the type of objects (e.g., supermarket items, spatial objects, sequences etc.), its associated attributes (e.g., weight, quantity etc.), and variant-specific constraints (e.g., how to generate candidates). Section 4 discusses this in detail.
- **What are the strengths and limitations of the framework?** The "power" of the framework depends on the capabilities of the rule mining engine and the variant description language. In Section 5, we provide an informal evaluation of the two components to give the reader an idea on the extensiveness of the framework.

2 Framework for Mining Rules

We begin with the question on what a generic engine for mining variants of rules can be. Our proposal centers around two key components: the algorithmic engine, and the variant description language.

2.1 Algorithmic Engine (VI)

Our observation on the current state of algorithms for mining rules shows that most of them are variations of the Apriori model. Efficient techniques (e.g., (Das, Ng & Woon 2001, Han & Pei 2000, Park, Chen & Yu 1997)), though available, are restricted to "plain-vanilla" rules. Since most are Apriori-based, we have taken the intuitive approach to derive the skeleton framework. The methodology is to identify the common aspects of each variant and decouple their differences.

We noted that the Apriori approach to finding frequent itemsets can be generalized into six steps; namely, (1) prepare the input to the algorithm, (2) generate candidates from the input, (3) remove candidates that are confirmed infrequent, (4) determine if a transaction supports a candidate; if so, (5) identify the support contribution, and finally (6) select candidates satisfying the support threshold. The frequent candidates then become the seed for the next pass. With this observation, we formulate the VI algorithm as shown in Figure 1.

```
Algorithm VI($I$: set of candidate 1-itemsets, $\varphi_{min}$: minimum support)
begin
    $L_p = \varnothing; k = 1; I_k = I;$
    do
        foreach $i \in I_k$ do
            $L_k = L_k \cup \text{MineItemsets}(i, L_p, \varphi_{min});$
        $L_p = L_k; k{+}{+}; I_k = L_1;$
    until $L_p = \varnothing;$
end

procedure MineItemsets($i$: candidate, $L_p$: frequent ($k-1$)-itemsets, $\varphi_{min}$: minimum support)
begin
    $L_f = \varnothing;\ S_k = \texttt{CandidateGen}(i, L_p, S_{k-1});$
    foreach $T \in$ Transaction-DB do
        foreach $s \in S_k$ do
            if ($\texttt{Support}(T, s)$)
            then $\texttt{Count}(s) \mathrel{+}= \texttt{GetSupport}(s, T);$
        endfor
    endfor
    foreach $s \in S_k$ do
        if ($\texttt{Satisfy}(\texttt{Count}(s), \varphi_{min})$) then $L_f = L_f \cup \{s\};$
    endfor
    return $L_f;$
end
```

Figure 1: The **V**ariant-**I**ndependent (VI) algorithm.

We have consciously omitted the evaluation of confidence for a rule since the generalization of that follows directly with the discovery of frequent itemsets. Having said that, the reader may note that an item refers to any concept such as a supermarket product, a spatial object or a sequence. In the simplest case, the input are atomic items representing candidate 1-itemsets. Since L_p is initially empty, the support of each candidate 1-itemset is collected. The 1-itemsets are then evaluated and forms the seed for generating candidates in the next pass. Notice that the frequent (k-1)-itemsets obtained at the end of each pass are stored in L_p and iteratively extended with the frequent 1-itemsets. Therefore, VI discovers all frequent itemsets in the database with respect to the input.

A slight deviation from the Apriori is the way we have structured the framework. To cover the entire problem space, the number of passes through the database depends on the size of the largest frequent itemset and the number of frequent 1-itemsets. This may raise concerns about performance. Nevertheless, the speed penalty may be minimized or made insignificant as discussed in Section 4.2. There are two factors that motivated the deviation: the intuitive integration with the work of (Johnson, Lakshmanan & Ng 2000, Ng et al. 1998, Ng et al. 1999) and the ease of describing variant-specific behaviors (discussed in the next section).

From an analyst's standpoint, the process of finding insights usually involves a section of the problem space and a subset of the database. Extending Example 1, the analyst may be interested in the sales transactions for the past one year (a subset of the database) containing dairy items (a subset of the problem space) where the sub-total for the dairy items in each transaction is less than $10 (i.e., the mining constraint). Existing algorithms do not structure naturally to support such goals. Instead, most attempt to cover the entire problem space leading to unfocused and time consuming computations. Recognizing this, we developed VI with consideration for goal-oriented tasks.

Back to Figure 1, notice that the core of VI is the **MineItemsets** procedure. Instead of covering the entire problem space, its input attribute allow the analyst to specify the subject of interest (the candidate x), the problem space to cover (considering only certain itemsets, frequent or infrequent) and the threshold (φ_{min}) to operate for the current pass. Hence,

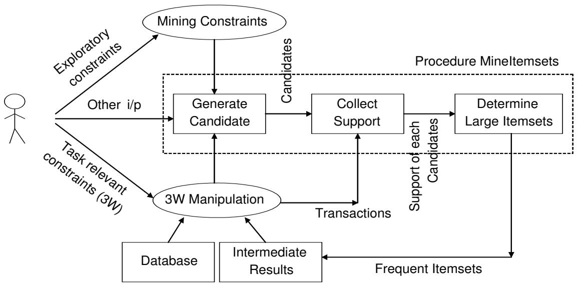


Figure 2: Focused mining of maximum knowledge.

MineItemsets can better integrate constrained mining mechanisms such as those proposed by Ng et al. (Ng et al. 1998, Ng et al. 1999) and facilitate manipulation at a higher level using implementations such as the **3W** model (Johnson et al. 2000). More importantly, **MineItemsets** is variant-independent and allows different attributes to be considered to mine informative rules. Set in this context, we believe the analyst at the supermarket will be more likely to achieve his goal with informative rules that provide deeper insights to the task relevant data.

As a concluding note to this section, Figure 2 illustrates, graphically, how mining constraints and the **3W** implementation inter-operate with **MineItemsets** to achieve user focused mining of maximum knowledge. For the supermarket analyst, the input to the mining constraints would be dairy items whose sub-total is less than \$10. To select only transactions for the past one year, he uses the **3W** operators. Finally, the support threshold and variant-specific constraints are entered directly into **MineItemsets**. The three sets of input thus control the candidates to be generated. The feedback of the frequent itemsets closes the loop to ensure goal focused discovery of the rules. Notice that VI does not operate on its own. The VDL, discussed next, provides the mechanism to describe variant-specific behaviors.

2.2 Expressing Variants of Rules

Viewed as a black box, we ask **MineItemsets** the question: "Which itemsets in L_p, with x considered, are frequent?". The answer depends on the six steps of mining frequent itemsets. Since step 1 is the input from the mining constraints and **3W** operations, addressing the remaining steps will provide the answer. From Figure 1, each step is represented as a function point in the algorithm. The task of expressing variant-specific behaviors therefore reduces to describing how the function points (i.e., `CandidateGen()`, `Support()`, `GetSupport()`, `Satisfy()`) should behave.

We first introduce a variant description language (VDL) to facilitate specification of attribute specific behaviors. Figure 3 shows the list of symbols and its semantic in the context of the VDL. The syntax of the VDL is shown below. For clarity and compactness, we used mathematical constructs in this section and the next. We then consider the users' standpoint and propose a VDL implementation that is SQL-like.

define [*variant-name*]

generate-cand by [*expression*] **prune if** [*expression*]

select-cand if [*expression*] **compute-cnt by** [*expression*]

select-freq if [*expression*]

The label *variant-name* uniquely identifies a definition while *expression* is a mathematical construct that describes the specifics of a particular variant. The "generate-cand by" and "prune if" defines the behavior of `CandidateGen()`. The "select-cand if" specifies how `Support()` is evalutated. The contribution of support is expressed in "compute-cnt by" for `GetSupport()`, and the behavior of `Satisfy()` is dependent on the definition of "select-freq if" in the VDL. Due to space constraints, we discuss an example of how variant-specific behaviors are defined using the above. Figure 4 contains the description of other variants and are elaborated in (Ong, Ng & Lim 2002*a*).

Symbol	Description
$\mathfrak{f}$	a frequent 1-itemset discovered from the seed of the algorithm
L_p	the set of frequent itemsets discovered in the previous pass
C	the candidate itemset generated by extending $\mathfrak{f}$ and L_p; $c \in C$ and c_i is i^{th} element of C
S_k	the candidate itemsets of size k before any pruning by `CandidateGen()`
T	a transaction in the database
φ_{min}	the minimum support threshold
$f(s)$	f is an attribute of s and $f(s)$ returns the attribute value of f in s (e.g., `MSB`(s))
$\times_a(\mathfrak{f}, L_p)$	`SortLex`$(\{\{\mathfrak{f}\} \cup x \mid x \in L_p\})$, the cross of two itemsets where items are ordered lexicographically (`SortLex`()) and duplicates removed (by definition of $\cup$)
$\times_r(\mathfrak{f}, L_p)$	$(\mathfrak{f} \times_a L_p) \cup (\{\oplus_{i=1}^{\lvert x\rvert+1} \mathfrak{f}\}) \cup$ `SortLex`$(\{\mathfrak{f} \oplus x \mid x \in L_p\})$, the cross of two itemsets where items are lexicographically ordered and duplicates retained ($\oplus$ operator)

Figure 3: Symbols and its semantic in the context of the variant description format (VDL).

Example 3 (Mining Rules with Multiple Concepts) One of the attributes that we can include is a concept taxonomy to mine items at multiple concept levels. As observed by Han and Fu (Han & Fu 1995), mining at the atomic level generates too many rules that are unlikely to be strong due to the small average support of each item within a large itemset. Hence, a domain analyst may introduce a concept taxonomy to mine informative rules using a higher level of abstraction. In the context of CrystalBall, this is done by writing the VDL defined as follows.

define `MultiLevelRules`

generate-cand by $\mathfrak{f} \times_a L_p$

prune if $\exists c \in C, C - \{c\} \notin L_p$

select-cand if $C \subseteq$ `GetConcept`(T, ℓ)

compute-cnt by 1

select-freq if `Count`$(C) \geqslant \varphi_m in$

In the above, the analyst enters a set of concept items (not atomic items) and define `GetConcept()` to map atomic items in the database. In the goal driven scenario, the input is the concept items and `GetConcept()` may be defined using the VDL (likewise for `Weight`, `Qty`, `MIS`, and `MSB`). Together with Figure 1, we see that the definition of a candidate may be produced by combining an item ($\mathfrak{f}$) with a list of itemsets (L_p) found frequent in the previous pass. Once generated, it is evaluated using the criteria defined in the "prune if" part of the VDL. The expression in "prune if" of Example 3 specifies the anti-monotone property as observed in the Apriori algorithm. If the candidate is not pruned, it will

Variant	VDL Description
Apriori/ Spatial Rules	**define** conventional-apriori **generate-cand by** $\mathfrak{f} \times_a L_p$ **prune if** $\exists c \in C, C - \{c\} \notin L_p$ **select-cand if** $C \subseteq T$ **compute-cnt by** 1 **select-freq if** $\texttt{Count}(C) \geqslant \varphi_m in$
Rules with Recurrent Items	**define** recurrent-rules **generate-cand by** $\mathfrak{f} \times_r L_p$ **prune if** $\exists c \in C, C - \{c\} \notin L_p$ **select-cand if** $C \subseteq T \wedge \forall c \in C, \texttt{Qty}(c) \leqslant \texttt{Qty}(T.c)$ **compute-cnt by** $\min(\{\lfloor \frac{\texttt{Qty}(T.c_i)}{\texttt{Qty}(c_i \in C)} \rfloor\}_{i=1}^{\lvert C \rvert})$ **select-freq if** $\texttt{Count}(C) \geqslant \varphi_m in$
Rules with Weighted Items	**define** weighted-rules **generate-cand by** $\mathfrak{f} \times_a (S_{k-1} -$ $\{s \in S_{k-1} \mid \texttt{Count}(s) < \texttt{MSB}(s)\})$ **select-cand if** $C \subseteq T$ **compute-cnt by** $\texttt{Count}(C) \times \sum \texttt{Weight}(c \in C)$ **select-freq if** $\texttt{Count}(C) \geqslant \varphi_{min}$
Rules with Multiple Minimum Support	**define** multi-support-rules **generate-cand by** $\{$ $\{c_i, c_j\} \mid c_i, c_j \in \{$ $y_p \in \{$ $i_1, i_2, \ldots, i_z \mid \forall(m < n) \in [1..z],$ $\texttt{MIS}(i_m \in I_1) \leqslant \texttt{MIS}(i_n \in I_1)$ $\} \mid \forall(k < q), \texttt{Count}(y_k) \leqslant \texttt{MIS}(y_q) \wedge$ $\texttt{Count}(y_q) \geqslant \texttt{MIS}(y_p) \wedge \texttt{Count}(y_p) \geqslant \texttt{MIS}(y_q)$ $\} \mid (i < j) \wedge (\lvert x \in L_p \rvert = 1) \wedge$ $\texttt{Count}(c_j) \geqslant \texttt{MIS}(c_i) \wedge \texttt{Count}(c_i) \geqslant \texttt{MIS}(c_i)$ $\} \cup \{\mathfrak{f} \times_a L_p \mid 1 \neq \lvert x \in L_p \rvert\}$ **prune if** $\exists c, c_1, c_2 \in C,\ C - \{c\} \notin L_p \wedge$ $((c_1 \in C - \{c\}) \vee (\texttt{MIS}(c_1) = \texttt{MIS}(c_2))$ **select-cand if** $C \subseteq T$ **compute-cnt by** 1 **select-freq if** $\texttt{Count}(C) \geqslant \min(\texttt{MIS}(c \in C))$
Sequences	**define** sequence-rules **generate-cand by** $\{$ $\langle f_1, \ldots, f_{\lvert\mathfrak{f}\rvert} \cup \{ x'_{\lvert x \rvert}\}\rangle \mid (\lvert\mathfrak{f}\rvert = \lvert X \rvert) \wedge$ $(f' \in f \in \mathfrak{f}) \wedge (x' \in x \in X \in L_p) \wedge$ $\langle f_1 - \{ f'_1\}, \ldots, f_{\lvert\mathfrak{f}\rvert}\rangle = \langle x_1, \ldots, x_{\lvert X \rvert} - \{ x'_{\lvert x \rvert}\}\rangle$ $\} \cup \{$ $\langle f_1, \ldots, f_{\lvert\mathfrak{f}\rvert}, \{x'_{\lvert x \rvert}\}\rangle \mid (\lvert\mathfrak{f}\rvert < \lvert X \rvert) \wedge$ $\langle f_1 - \{ f'_1\}, \ldots, f_{\lvert\mathfrak{f}\rvert}\rangle = \langle x_1, \ldots, x_{\lvert X \rvert} - \{ x'_{\lvert x \rvert}\}\rangle$ $\}$ **prune if** $\exists(c' \in c) \in C, C.c - \{c'\} \notin L_p$ **select-cand if** $\forall c \in C, c_i \subseteq (t_i \in T)$ **compute-cnt by** 1 **select-freq if** $\texttt{Count}(C) \geqslant \varphi_{min}$

Figure 4: VDL description for other variants.

be checked against the transactions in the database. We determine if a transaction supports the candidate by the criteria given in the "select-cand if" of the VDL. In the example, `GetConcept()` translates the atomic items in the transaction to the desired level ℓ to facilitate comparison. If the subset property holds, the engine determines the support count contributed by evaluating the expression in "compute-cnt by", which in this case, is simply 1. Finally, after the pass through the database, VI evaluates each itemset (based on "select-freq if"), returning those that are frequent.

Of course, the existing VDL description is not appropriate from the users' standpoint. The syntax is awkward and is difficult to express via the keyboard. At the same time, the mathematical constructs are also difficult to interpret programmatically. Considering the users' standpoint, the VDL should be implemented like any 4GL languages. Recently, we proposed a SQL-like implementation of the VDL (Ong, Ng & Lim 2002*b*) which is shown below. This VDL describes the same task of mining plain association rules as depicted mathematically in Figure 4.

```
GENERATE conventional-apriori USING
CANDIDATES FROM AprioriJoin(f, Lp)
PRUNE IF EXISTS C - C.c NOT IN Lp
VOTE IF Subset(C, T)
INCREMENT C.Count BY 1
SELECT IF C.Count >= MinSupp
```

Back to Example 3, we see that the VDL specifies the behavior of VI at the itemset level. In our initial work, we operated on a set of itemsets leading to complex specification and poor performance. Going down to the itemset level simplifies the VDL and admits efficient in-lining of the expressions with the algorithmic engine. Also, the reader may note the optional use of the "prune if" clause in the VDL as illustrated in the mining of weighted items[1]. Of course, the ability to describe each attributes alone is insufficient. A domain analyst can actually combine different attributes easily in CrystalBall to mine informative rules. We devote the next section to this discussion.

3 Mining Informative Rules

So far, we have described some existing variants using the VDL. In this section, we show how new variants can be created by combining different attributes into a coherent whole for mining informative rules. Rather then explaining the procedure, we illustrate the process with the two examples discussed earlier.

3.1 Mining Supermarket Data

The analyst in Example 1 considered two attributes in the rules to extract: the weight and the recurrence of an item. Using CrystalBall, the analyst's goal is to write the VDL to be interpreted by the algorithmic engine, VI. The first step is to determine how candidates should be generated. Since we are building on existing attributes, we derive the candidate generation procedure from Figure 4.

In the weighted case, candidates are obtained by extending the candidate itemsets from the previous pass with $\mathfrak{f}$ (i.e., $\mathfrak{f} \times_a S_{k-1}$). In the case of mining recurrent items, the VDL specifies that candidates are generated by $\mathfrak{f} \times_r L_p$. Since the candidates are all possible patterns that can occur in the database, the union of their supersets is the set of all candidates that we need to consider in our example.

Observation 1 From Figure 3, $\mathfrak{f} \times_r L_p$ is actually $(\mathfrak{f} \times_a L_p) \cup \{\{\oplus_{i=1}^{|x|+1}\mathfrak{f}\}\} \cup \{\{\mathfrak{f} \oplus x \mid x \in L_p\}\}$. Since $\times_r$ is an extension of $\times_a$, then $(\mathfrak{f} \times_a L_p) \subseteq (\mathfrak{f} \times_r L_p)$. Likewise, $S_{k-1} \supseteq L_p$ and thus, the candidate generation is specified as $\mathfrak{f} \times_r S_{k-1}$.

In this example, the analyst would probably skip the pruning criteria since his goal is focused (i.e., transactions for the last year containing diary items whose sub-total is less than \$10). In such cases where the problem space is small, it is more productive to collect all the support counts in a single scan of the data. However, for the purpose of illustration, we discuss how he derives the expressions for pruning candidates.

Observation 2 The mining of weighted items invalidates the anti-monotone property as each weight alters the final support. As a result, the pruning criteria of recurrent items (with strong dependence on anti-monotone) cannot be used.

[1] Since by definition, "prune if" evaluates a candidate after it has been generated, the posterior pruning in the algorithm proposed by Cai et al. has to be expressed in "generate-cand by" instead to maintain the anti-monotone property.

Instead, the posterior pruning using the minimum support bound (i.e., $\mathtt{MSB}$) is applied to maintain the anti-monotone property so that Apriori-like performance is sustained. Thus, there will be no "prune if" clause and "generate-cand by" is extended to $\text{ʃ} \times_r (S_{k-1} - \{\, s \in S \mid \mathtt{Count}(s) < \mathtt{MSB}(s)\})$ instead.

Comparing the criteria for selection in Figure 4, we see that the recurrent definition has a stricter criteria over the weighted one. Unlike the notion of a weight of an item, the quantity of an item influences a transaction in supporting an itemset.

Observation 3 Considering both attributes imply the conjunction of the attributes' criteria, which reduces to $(C \subseteq T) \wedge (\forall c \in C, \mathtt{Qty}(c) \leqslant \mathtt{Qty}(T.c))$.

The next step is to determine the support count. Each item has an associated weight that alters the raw support count obtained from the database. Depending on the context and requirement, if an item occurs twice in the itemset, the analyst may consider the contribution of the item weight twice or simply once. Either way, the support count will be altered differently.

Observation 4 Considering recurrences in the total weight of an itemset, we compute the final support count for this scenario by substituting the raw support computed in the recurrence case as the raw support of the weighted itemset. Hence, the support of an itemset is $\min(\{\lfloor \frac{\mathtt{Qty}(T.c_j)}{\mathtt{Qty}(c_j \in C)} \rfloor_{j=1}^{|C|}\}) \times \sum \mathtt{Weight}(c \in C)$.

The last step (to select candidates that are frequent) is relatively straightforward. A check with Figure 4 shows that all have the same evaluation criteria. Thus, the criteria for "select-freq if" follows. Putting them together, we obtain the final VDL as shown below.

```
define informative-rules-for-example-1
generate-cand by $\text{ʃ} \times_r (S_{k-1} - \{s \in S \mid \mathtt{Count}(s) < \mathtt{MSB}(s)\})$
select-cand if $(c \subseteq T) \wedge (\mathtt{Qty}(c) \leqslant \mathtt{Qty}(T.c))$
compute-cnt by $\min(\{\lfloor \frac{\mathtt{Qty}(T.c_j)}{\mathtt{Qty}(c_j \in C)} \rfloor_{j=1}^{|C|}\}) \times \sum \mathtt{Weight}(c \in C)$
select-freq if $\mathtt{Count}(c) \geqslant \varphi_{min}$
```

3.2 Mining Abstract Spatial Objects

In Example 2, we moved from supermarket items to spatial objects and considered two other attributes: high level concept and multiple minimum support (a variation of weighted items). From Example 1, the analyst observes that the creation of variants in CrystalBall is semi-mechanical and rationalize as follows.

- The set of candidates generated by a particular variant, or the union of candidates produced by different variants is the superset that provides a complete cover of the composition's problem space.
- A candidate cannot be pruned unless it is known to be infrequent in the coming passes with all attributes considered.
- Testing if a transaction supports a candidate requires that it satisfies all attributes under consideration. As such, the strictest criteria is used. This also applies to the evaluation of a candidate to determine its eligibility as frequent.
- Support counting is done in two steps: derive the lowest raw support count and then consider the attributes that alter this value.
- If a particular clause has the same criteria for all attributes considered, the composition follows.

Using the above observations, the analyst derives the following VDL. In the same way, we can explain the VDL using these considerations.

```
define informative-rules-for-example-2
generate-cand by $\{\{c_i, c_j\} \mid c_i, c_j \in \{\, y_p \in \{$
        $i_1, i_2, \ldots, i_z \mid \forall(m < n) \in [1..z],$
                        $\mathtt{MIS}(i_m \in I_1, \ell) \leqslant \mathtt{MIS}(i_n \in I_1, \ell)$
    $\} \mid \forall(k < q), \mathtt{Count}(y_k) \leqslant \mathtt{MIS}(y_q, \ell) \wedge$
        $\mathtt{Count}(y_q) \geqslant \mathtt{MIS}(y_p, \ell) \;\wedge\; \mathtt{Count}(y_p) \geqslant \mathtt{MIS}(y_q, \ell)$
    $\} \mid (i < j) \;\wedge\; (|x \in L_p| = 1) \;\wedge\; \mathtt{Count}(c_j) \geqslant \mathtt{MIS}(c_i, \ell) \wedge$
        $\mathtt{Count}(c_i) \geqslant \mathtt{MIS}(c_i, \ell)$
    $\} \cup \{\, \text{ʃ} \times_a L_p \mid 1 \neq |x \in L_p|\}$
prune if $\exists c, c_1, c_2 \in C,\ C - \{c\} \notin L_p \wedge$
            $((c_1 \in C - \{c\}) \vee (\mathtt{MIS}(c_1, \ell) = \mathtt{MIS}(c_2, \ell))$
select-cand if $C \subseteq \mathtt{GetConcept}(T, \ell)$ compute-cnt by $1$
select-freq if $\mathtt{Count}(c) \geqslant \min(\mathtt{MIS}(c \in C, \ell))$
```

The input contains spatial objects at the concept level ℓ as designated by the analyst. At the same time, `MIS()` has been extended to return the individual support of a spatial object at a given concept level. By the first observation, candidates generated for multiple minimum support covers the problem space of the composition. With the extension of `MIS()`, the pruning criteria considers both attributes before discarding a candidate as determined in the second observation. In the third observation, a transaction can only support a candidate if it satisfies the attributes considered. Since multiple support thresholds do not affect the evaluation, the only constraint is to import each transaction at the appropriate concept level. This is done using the "select-cand if" criteria in Example 3. Using the last observation, the support count is 1. Finally, a candidate is determined frequent by the smallest support threshold in the itemset at concept level ℓ (again by the third observation).

From the above examples, we illustrated the value proposition of combining existing variants to create informative rules. More importantly, CrystalBall is not limited to the existing variants. It is possible to consider new attributes, implement the mechanism using the VDL, and later, combined with existing variants to create other types of association rules.

4 Experimental Studies

The objective of this study is to develop a proof of concept on the framework's feasibility. We then compare how the framework approach performs with respect to optimum implementations. This is to assess the overheads (a concern for some readers) involved and to path the ground work for optimization.

4.1 Methodology

We conducted the experiments on a single-CPU Pentium-3 workstation at 1.7 GHz, cache size 256KB, and 256MB of RAM having Windows 2000 as the operating system. We used text files as the database (where each line represents a transaction), the .Net runtime to execute the code, and the new C# programming language to implement each variant and the framework.

For a fair and realistic comparison, each variant closely implements the algorithm suggested by the respective authors. We then run tests on each variant with different support thresholds and database parameters (i.e., the number of items, transactions, patterns etc). The results are then compared to instances created from CrystalBall. For each variant, we instantiated an equivalent using the VDL in Figure 4 and benchmarked their performance. We next instantiated two algorithms from CrystalBall to realize the scenarios described in the examples earlier. Since we are unaware of similar priori work, the tests performed here served as a relative measure of how the algorithms scaled with the inclusion of additional attributes under different experimental settings.

In the first example, we first generate synthetic transactions using the program from the IBM QUEST site (*IBM QUEST: The Data Mining Group* n.d.). Each transaction is then modified by altering the recurrence value of certain items while pruning others to maintain its average size. This is done by first selecting the item in the transaction with the smallest weight. A random positive number m, bounded by the `MaxOccur` (Zaiane et al. 2000) value, is assigned as its occurrence in the transaction. Next, $m-1$ other items are selected at random from the transaction and their occurrences decremented by 1. This is repeated for the next smallest weighted item up to the number of items to be modified for the current transaction. Items with occurrence value less than 1 are pruned and the transaction written to disk. The weight of each item is generated using a normal distribution with a mean of 0.5 and are assigned randomly.

In the second example, the synthetic transactions are spatial objects instead of items, and are generated using the same synthetic data generator. The concept-tree is then constructed based on the database and two given attributes: the average fanout f of each node and the number of levels ℓ of the concept tree. Using f and ℓ, the number of leaves in the tree is determined and each spatial object is assigned randomly to one of the leaf node. The individual support threshold is generated in the same way as the weight in the first example except that the mean of the normal distribution is now user specified.

To instantiate a variant from the framework, we invoke the VDL interpreter described in Section 2.2 to translate the VDL into C# code and then emit the executable by invoking the C# compiler. The instance is then ran against the optimum implementation with the same test attributes to obtain a comparison of their results. Each instance created from the framework uses a database cache to compensate the extra passes through the database as noted in Section 2. As a result, the physical I/O cost incurred in our test is minimized, and eliminated in cases where the entire database can be loaded into memory[2]. The details are discussed below.

4.2 Discussions

We begin with Figure 5(a) which shows the relative performance of the Apriori against an instance created from CrystalBall under different test parameters. In Figure 5(a), the CrystalBall instance performed better than its optimum counterpart except for higher thresholds in the third test parameter (i.e., T25N10KD100K). This speedup can be attributed to optimizations in CrystalBall that overcome the overheads in the VDL. Specifically, the amount of overheads in the VDL can be estimated by the number of "set" operations. Therefore, the Apriori variant has the lowest overheads as seen in Figure 4.

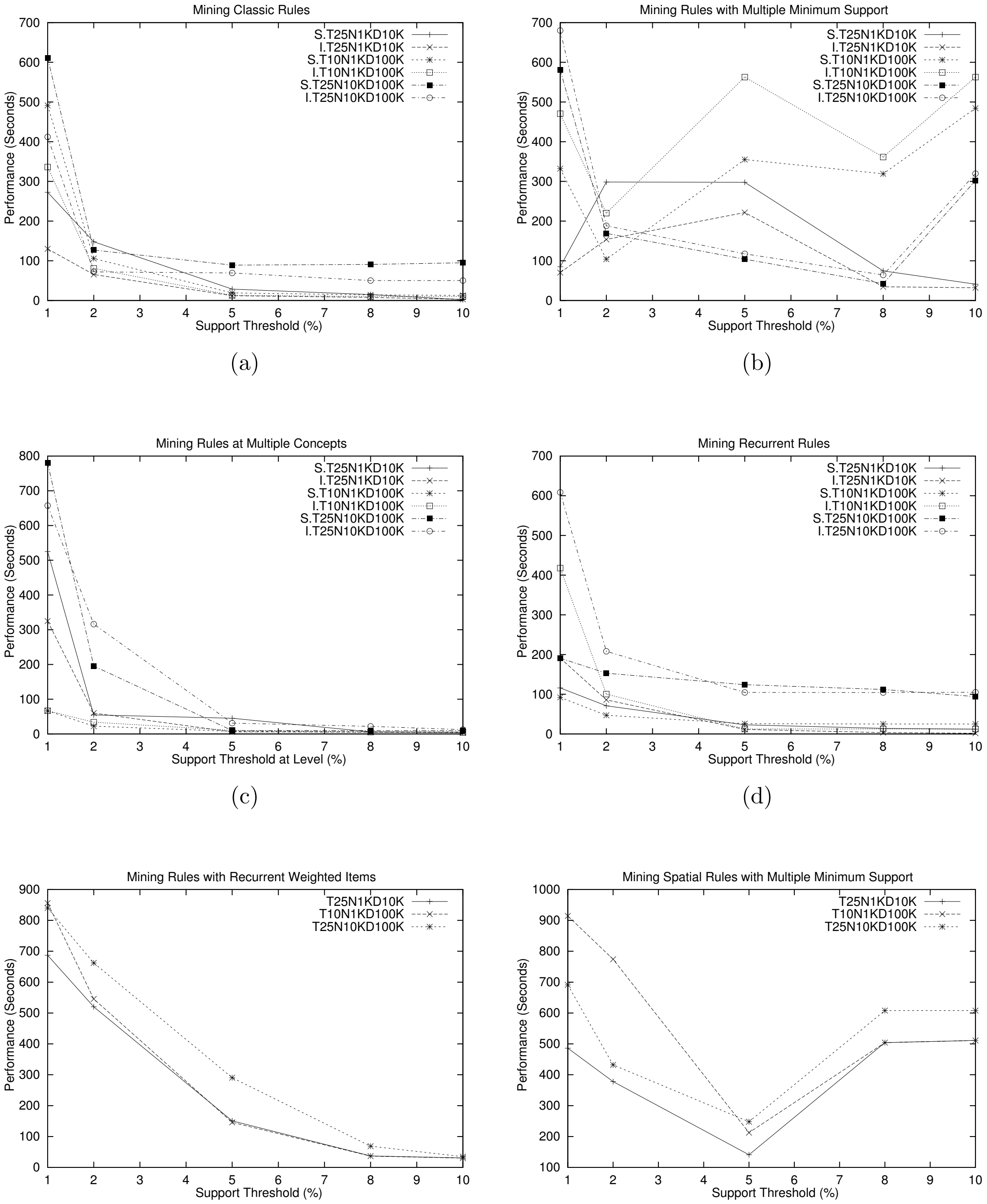
Figure 5(b) shows the relative performance of the MSApriori and its CrystalBall instance. Unlike the Apriori, where a single support is used, the items in MSApriori are given individual minimum support (i.e., `MIS`). As such, the "support" in Figure 5(b) does not translate to the "support" in the Apriori sense, but refers to the average `MIS` value of each item. Depending on the combination of items in each transaction, and the way the MIS values are generated synthetically, the runtime results do not exhibit a logarithmic-like curve similar to other variations. For example, the average support at 10% has more itemsets than the average MIS support at 8% and 2%. This is due to the selection of an itemset as frequent by the smallest MIS value that exists within a candidate itemset (see "select-cand if" clause of multiple minimum support rules in Figure 4). From the results, we see that the overheads incurred by the VDL is now more than the improvements made by the optimization. This can be explained by Figure 4 where the VDL for multiple minimum support is more complicated than that of the classic Apriori.

Even a simple VDL can result in poor performance if the VDL compiler is not optimizing (which is the case in our prototype implementation). This happens when we compiled the VDL for mining multiple concept rules. The initial performance of the CrystalBall instance was more than a magnitude slower than the specific implementation. Investigation shows that repeated translation (by `GetConcept()`) is computationally expensive and the condition deteriorates as the database grows larger. We overcome this by caching the translated transaction instead of its atomic version giving the results seen in Figure 5(c). At higher support thresholds, both instances has roughly the same runtime. However, the overheads introduced by the translation (due to `GetConcept()`) becomes apparent at lower supports where the number of candidates increases.

The fourth graph (Figure 5(c)) compares the performance of mining rules containing recurrent items. The optimum implementation uses the MaxOccur algorithm (Zaiane et al. 2000) and the CrystalBall instance is created from the VDL described in Figure 4. Interestingly, the performance of both implementations at the support of 2% onwards remain relatively consistent. This deviates from our normal expectation of how Apriori-like algorithms behave. The rationale behind the observation lies not in the algorithm, but the synthetic data generated. We want to maintain the average transaction size as we twinned the recurrences of selected items in the transaction. Since items need to be removed after twinning, the data condition is altered and causes support of itemsets to exist at low frequency.

The last two graphs shows the absolute performance of two compositions that mine rules described in Example 1 (Figure 5(e)) and Example 2 (Figure 5(f)). We tested the performance with similar parameters used in the first four tests but are synthetically altered for each composition. Since compositions has more constraints and operations, the performance of finding the frequent itemsets is expected to be longer. Using the **3W** model as a relative measure, finding rules in Example 1 requires a sequential run of two algorithms, one to find weighted rules and the other to discover recurrent items, before it can be manipulated in **3W**. In CrystalBall, this requires

[2] When necessary, the structure of the framework can be modified to make the same number of passes as the Apriori without any changes to the VDL definition.



(a)

(b)

(c)

(d)

(e)

(f)

Figure 5: (a) - (d): Relative performance of CrystalBall against implementations including Apriori, FastApriori, MSApriori (multiple support thresholds), MaxOccur (recurrent items) and MLApriori (multiple concepts); (e) & (f): Performance of compositions created using CrystalBall for Example 1 and 2. Smaller values on the Y axis indicates better performance. Each test is described using {S, I}.Tx Ny Dz where S represents the optimum implementation and I, the CrystalBall instance of S; x the average size of each transaction; y the number of unique items (or spatial combinations in the case of Example 2), and z the number of transactions in the database.

only one pass through the data, and all parameters are pushed deep into the process and evaluated simultaneously. Although we do not conduct empirical tests for this, CrystalBall's approach should, intuitively, provide better performance.

Overall, the framework performed consistently below that of its optimized cousins. This is expected due to the abstractions involved and our focus on feasibility rather then performance, the future work to be addressed next.

5 Framework Evaluation

We have, in the previous section, demonstrated the feasibility of the framework and included a brief discussion on its performance. For completeness, we present an informal evaluation of the framework, as an extensive and formal review would be difficult within the limits of the paper.

5.1 Algorithmic Engine

The comprehensiveness of the framework depends on the algorithmic engine and the VDL. Currently, the generic rule mining engine uses the Apriori algorithm as its foundation. Comparing this against approaches such as the `FP-Growth`, we see that the balance in performance and the generic approach inherent in its design is what motivated work in other variants to be based on the Apriori. The generate-and-test approach, although inefficient in the current state-of-the-art, is generic and intuitive for considering different attributes. The task of candidate generation is to enumerate all patterns in the problem space, and then evaluate (i.e., candidate test) if it holds for the given data. This makes it easy to specify what are the possible patterns, and how to evaluate the patterns in the context of different attributes.

For example, in Figure 4, we see that an attribute p (e.g., weight) can be considered step by step through the specification of constraints in the VDL to define its behavior during mining. This attribute-focused approach, we believe, should provide a sufficiently general framework for describing new attributes beyond what has been studied in this paper. Above all, the same approach is what makes the mining of informative rules (in a single run of the data mining task) possible.

However, the strength of the Apriori is also its weakness. The generate-and-test process incurs costly computations. According to Han and Plank (Han & Plank 1996), the algorithm has dependency on many parameters including the amount of memory, the number of transactions, candidates, frequent itemsets, and the length of a frequent itemset. This meant that the Apriori has varying performance under different environments and data conditions (which is also observed in our tests, see Figure 5)). Many techniques has been introduced to improve Apriori (Agrawal & Srikant 1994, Das et al. 2001, Hipp, Güntzer & Nakhaeizadeh 2000, Ng et al. 1998, Ng et al. 1999, Park et al. 1997), and some were used to enhance the performance of the framework.

The other limitation of Apriori is in the type of rules it can discover. In our current framework, we are only capable of mining intra-transaction rules. The algorithm does not considers situations such as mining rules from inter-transactions (Tung, Lu, Han & Feng 1999), multiple relations (Dehaspe & Raedt 1997), or rules containing continuous variables (Srikant & Agrawal 1996). These variants, when considered with different attributes, are equally informative and important. As an initial work, we have chosen the Apriori as our starting point. To ensure extensiveness and comprehensiveness of the framework, future work in improving the data mining engine will consider the issues mentioned above.

5.2 Variant Description Language

Evaluating the data mining engine provides a gauge on the domain of rules that the current proposal is capable of. Specifically, our evaluation of the algorithmic engine reveals the ability to perform mining of intra-transaction rules. Within this domain, the extensiveness of intra-transaction rules are then determined by the VDL.

The formal model of the VDL is the use of set theory and first order logic as seen in the expressions in Figure 4. Using the notion of a set of items, the VDL closely models the underlying sub-system of the Apriori architecture and that of a transaction database. Adding first order logic predicates, different attributes of each item relative to the database, transaction, or any other entities may be retrieved. This gives rise to the possibilities of attributes that can be associated with an item and potentially, beyond those discussed in the paper. An item, in the context of this paper, is a concept that may represent a physical product from the supermarket, a spatial object in the spatial database, or a sequence of events in an event log. A transaction is therefore a set of such items that occurs with respect to certain events (e.g., a purchase) or concepts (e.g., an image) that relates the items together.

The modelling of items and transactions is an important criteria towards the extensiveness of the framework. The freedom to operate across multiple domains and data sources gives good flexibility to the framework. At the same time, it also encourages the use of attributes associated with each domain to be used for finding informative rules. By providing an environment to consider these attributes, the framework also encourages the possibilities of forming new informative rules through composition of related attributes obtained across different domains. We believe that this approach will help introduce new attributes beyond those discussed in this paper.

From the samples given in Figure 3, we see that the framework makes available the intermediate results produced during a run of the data mining task. These intermediate results are made available for manipulation within the VDL. In addition, the VDL has contributions at various decision points in the framework, giving a fine granularity of control to the runtime behavior of the data mining engine. This gives the VDL good manipulative power when combined with the modelling of items and transactions discussed earlier.

The VDL is also expressive as demonstrated in the description of various association rules in Figure 4. Each function point is defined using first order logic based on the intermediate results obtained at the instance of evaluating the expression. At each function point, attributes related to an item in the context of the database or a transaction can be considered, evaluated, and altered in its value. More importantly, the interaction of attributes is limited only by the semantics of their composition and the objective of the domain analyst. In summary, the characteristics observed above should provide sufficient expressiveness in mining intra-transaction rules currently considered by the framework.

6 Related Work

The work in this paper was motivated from Mannila's (Imielinski & Mannila 1996, Mannila 1997, Mannila 2000) discussion on a theoretical framework for data mining. He commented on the ad-hoc situation of data mining research and called for a systematic framework to develop KDD applications. A framework for mining rules was discussed but lacked details to support its feasibility. Our work continues from where Mannila stopped. Although far from being a theoretical foundation for the mining of association rules, it serves well as a concrete basis for further work in this direction.

Mannila's vision for a theoretical foundation was realized at a high level by Johnson et al. (Johnson et al. 2000). The **3W** is a theoretical model for manipulating data mining tasks (algebraically) such that the output of one can be manipulated and marshalled to be the input of another. As illustrated in the paper, the effect of such operations create more insights that cannot be accomplished by individual tasks. In the existing paradigm, different variant of rules are discovered on its own by the respective algorithms. To discover informative rules within the **3W** model translates to the availability of these algorithms before the **3W** operators can be used to manipulate and achieve the same end result as CrystalBall. This dependency on the availability of the algorithms is not addressed by the model, not to mention the performance in practical situations where the tasks ran sequentially. Again, CrystalBall complements the **3W** implementation by an efficient mechanism to discover informative rules and eliminates the dependency on the availability of algorithms for variants of rules.

The `MINE RULE` operator proposed by Meo et al. (Meo, Psaila & Ceri 1996) enables a uniform and consistent description of the problem of discovering rules. SQL-like, the `MINE RULE` operator describes the different rule mining tasks for a subset of variants addressed in this paper. Meo's framework is concerned with the description of different tasks while our proposal, similar in motivation, describes how different attributes can be combined to discover the most informative rule from the database. Viewed from another perspective, the `MINE RULE` operator presents a consistent interface for describing the task of mining rules, and CrystalBall complements that with a consistent algorithmic engine for mining informative rules.

7 Conclusions

In this paper, our focus has been on the framework that eliminates the cost of engineering algorithms via a declarative approach to discover informative rules. We demonstrated the coverage of the framework on different variants of rules, and showed that the proposal is feasible with an implementation.

Our immediate future work is to address the performance aspects of CrystalBall. This is necessary to realize the vision laid out in the beginning of the paper, and to address emerging trends in active mining where data and user needs changes over time. Based on our preliminary findings, the decision to adopt the Apriori as the underlying model of our framework does not impede performance enhancements. Our initial results to scale the framework on large databases has been promising. In the optimum case, we managed to achieve a performance boost up to an order of magnitude over the Apriori algorithm and close to the `FP-Growth`. Our preliminary idea is as follows.

We construct a "transaction graph" that is a compact representation of the database. Unlike the `FP-Tree` that holds meta-data of the database, our approach has no loss of information and hence supports the mining of different variants without any modifications. Each node represents the unique item in the database and contains the global properties of the item (e.g., weight). The edges connecting the nodes has transaction IDs and transaction specific attribute values for each item it holds (e.g., quantity). Given an candidate C, the basic idea is to traverse *only* relevant edges and then perform a *selective* TID intersection to determine the collection of transactions that has the probability of supporting C. In other words, we "prune" transactions that guarantees no support for C, thus avoiding the overheads in scanning the entire database. This subset of transactions is the `Transaction-DB` scanned by algorithm VI in Figure 1. Hence, if C appears only in $i\%$ of the transactions, the traversal of our data structure will return at most $(i + j)\%$ of the transactions to be scanned by VI, where j is the percentage of transactions that appears to support C but failed to do so in reality. Since $i >> j$ (most of the time), the database scan is close to the actual support (i.e., $\approx i\%$) and the cost of traversing the data structure is only a fraction of the database. We shall report the details of this work in a future paper.

A survey on the landscape of data mining research reveals that the focus has been primarily on the algorithmic phases of the KDD process. In the context of rule mining, most algorithmic efforts are due to the consideration of new variants or the ability to outperform an earlier proposal. However, the value in doing so diminishes quickly as other aspects of KDD are weakly addressed and the analyst's goal lies in a section of the problem space. We therefore agree with (John 1999, Kohavi & Provost 2001) to call for a focus on other aspects of KDD to realize the goal of data mining. Some works has been observed recently (Bayardo & Agrawal 1999, Galhardas, Florescu, Shasha, Simon & Saita 2001, Raman & Hellerstein 2001) and CrystalBall's contribution in this aspect is to eliminate the need for engineering new algorithms. Given the engine and the variant-specification language, new attributes can be considered and compositions developed. This in turn eliminates programming, frees the analyst from the restrictions of the data mining application, and creates time for exploration of other KDD aspects.

Unlike other proposals, CrystalBall's focus is on deriving maximum information in the results. Hence, CrystalBall mines informative rules that previous proposals are incapable of. Our approach, we believe, will bring us back to where we started, i.e., to discover useful insights from data.

References

Agrawal, R. & Srikant, R. (1994), Fast Algorithm for Mining Association Rules, *in* 'Proc. of VLDB', Santiago, Chile, pp. 487–499.

Bayardo, R. J. & Agrawal, R. (1999), Mining the Most Interesting Rules, *in* 'Proc. of ACM SIGKDD', San Diego, CA, USA, pp. 145–154.

Cai, C. H., Fu, A. W. C., Cheng, C. H. & Kwong, W. W. (1998), Mining Association Rules with Weighted Items, *in* 'Proc. of IDEAS Symp.'.

Das, A., Ng, W.-K. & Woon, Y.-K. (2001), Rapid Association Rule Mining, *in* 'Proc. of CIKM', Atlanta, GA, USA.

Dehaspe, L. & Raedt, L. D. (1997), Mining Association Rules in Multiple Relations, *in* 'Workshop on Inductive Logic Programming', pp. 125–132.

Dong, G. & Li, J. (1998), Interestingness of Discovered Rules in Terms of Neighborhood-Based Unexpectedness, *in* 'Proc. of PAKDD', Melbourne, Australia, pp. 72–86.

Galhardas, H., Florescu, D., Shasha, D., Simon, E. & Saita, C.-A. (2001), Declarative Data Cleaning: Language, Model, and Algorithms, *in* 'Proc. of VLDB', Italy.

Han, J. & Fu, Y. (1995), Discovery of Multiple-Level Association Rules from Large Databases, *in* 'Proc. of VLDB', Zurich, Swizerland.

Han, J. L. & Plank, A. W. (1996), Background for Association Rules and Cost Estimate of Selected Mining Algorithms, *in* 'Proc. of CIKM', Rockville, MD, USA.

Han, J. & Pei, J. (2000), 'Mining Frequent Patterns by Pattern-Growth: Methodology and Implications', *ACM SIGKDD Explorations (Special Issue on Scalable Data Mining Algorithms)* **2**(2).

Han, J., Pei, J. & Yin, Y. (2000), Mining Frequent Pettern Without Candidate Generation, *in* 'Proc. of SIGMOD', Dallas, TX.

Hipp, J., Güntzer, U. & Nakhaeizadeh, G. (2000), Mining Association Rules: Deriving a Superior Algorithm by Analysing Today's Approaches, *in* 'Proc. of European Symp. on Principles of DMKD', France.

IBM QUEST: The Data Mining Group (n.d.), `http://www.almaden.ibm.com/cs/quest/syndata.html` .

Imielinski, T. & Mannila, H. (1996), 'A Database Perspective on Knowledge Discovery', *Communications of the ACM* **39**(11), 58–64.

John, G. H. (1999), 'Behind-the-Scences Data Mining: A Report on the KDD-98 Panel', *ACM SIGKDD Explorations* **1**(1).

Johnson, T., Lakshmanan, L. V. S. & Ng, R. T. (2000), The 3W Model and Algebra for Unified Data Mining, *in* 'Proc. of VLDB', Cairo, Egypt.

Kohavi, R. & Provost, F. (2001), 'Applications of Data Mining to Electronic Commerce', *Int. J. on DMKD* .

Liu, B., Hsu, W. & Ma, Y. (1999), Pruning and Summarizing the Discovered Associations, *in* 'Proc. of ACM SIGKDD', San Diego, CA, USA.

Liu, B., Hu, M. & Hsu, W. (2000), Multi-Level Organization and Summarization of the Discovered Rules, *in* 'Proc. of ACM SIGKDD', Boston, USA.

Mannila, H. (1997), Methods and Problems in Data Mining, *in* 'Proc. of ICDT', Delphi, Greece.

Mannila, H. (2000), 'Theoretical Frameworks of Data Mining', *ACM SIGKDD Explorations* **1**(2).

Mannila, H., Toivonen, H. & Verkamo, A. I. (1995), Discovering Frequent Episodes in Sequences, *in* 'Proc. of ACM SIGKDD', Montreal, Canada.

Meo, R., Psaila, G. & Ceri, S. (1996), A New SQL-like operator for Mining Association Rules, *in* 'Proc. of VLDB', Mumbai (Bombay), India.

Ng, R. T., Lakshmanan, L. V. S. & Han, J. (1998), Exploratory Mining and Pruning Optimizations of Constrained Association Rules, *in* 'Proc. of SIGMOD', Washington, USA.

Ng, R. T., Lakshmanan, L. V. S. & Han, J. (1999), Exploratory Mining via Constrained Frequent Set Queries, *in* 'Proc. of ACM SIGKDD', Philadelphia, USA.

Ong, K.-L., Ng, W.-K. & Lim, E.-P. (2002*a*), 'Mining Variants of Association Rules Using the CrystalBall Framework', CAIS Technical Report, `http://www.cais.ntu.edu.sg:8000/~ongkl/papers/tr.pdf` .

Ong, K.-L., Ng, W.-K. & Lim, E.-P. (2002*b*), VDL: A Language for Active Mining Variants of Association Rules, *in* 'Proc. of Int. Workshop on Active Mining (in conj. with IEEE ICDM)', Japan.

Park, J. S., Chen, M.-S. & Yu, P. S. (1995), An Effective Hash-Based Algorithm for Mining Association Rules, *in* 'Proc. of ACM SIGMOD', San Jose, CA, USA.

Park, J. S., Chen, M.-S. & Yu, P. S. (1997), 'Using a Hashed-Based Method with Transaction Trimming and Database Scan Reduction for Association Rules', *IEEE TKDE* **9**(5), 813–825.

Pei, J. & Han, J. (2000), Can We Push More Constraints into Frequent Pattern Mining?, *in* 'Proc. of ACM SIGKDD', Boston, MA.

Raman, V. & Hellerstein, J. M. (2001), Potters Wheel: An Interactive Data Cleaning System, *in* 'Proc. of VLDB', Roma, Italy.

Savasere, A., Omiecinski, E. & Navathe, S. B. (1995), An Efficient Algorithm for Mining Association Rules in Large Databases, *in* 'Proc. of VLDB', pp. 432–444.

Srikant, R. & Agrawal, R. (1996), Mining Quantitative Association Rules in Large Relational Tables, *in* 'Proc. of SIGMOD', Montreal, Canada.

Tung, K. H., Lu, H., Han, J. & Feng, L. (1999), Breaking the Barrier of Transactions: Mining Inter-Transaction Association Rules, *in* 'Proc. of ACM SIGKDD', San Diego, CA, USA.

Zaiane, O. R., Han, J. & Zhu, H. (2000), Mining Recurrent Items in Multimedia with Progressive Resolution Refinement, *in* 'Proc. of ICDE', San Diego.