Beyond Association Rules: Generalized Rule Discovery

Geoffrey I. Webb (webb@infotech.monash.edu.au)
School of Computer Science and Software Engineering,
Monash University,
Building 26, Melbourne, Victoria 3800 Australia
Telephone +61 3 9905 3296 Facsimile +61 3 99055146

Songmao Zhang (szhang@nlm.nih.gov)
National Library of Medicine (LHC/CgSB)
8600 Rockville Pike, MS 43 (Bldg 38A, B1N28T)
Bethesda, MD 20894 USA

Abstract. Generalized rule discovery is a rule discovery framework that subsumes association rule discovery and the type of search employed to find individual rules in classification rule discovery. This new rule discovery framework escapes the limitations of the support-confidence framework inherent in association rule discovery. This empowers data miners to identify the types of rules that they wish to discover and develop efficient algorithms for discovering those rules. This paper presents a scalable algorithm applicable to a wide range of generalized rule discovery task and demonstrates its efficiency.

Keywords: Rule Discovery, Association Rules, Classification Rules, Rule Search, Pruning Rules

1. Introduction

Rule discovery has evolved within two distinct paradigms. The earliest approaches to rule discovery (Buchanan and Feigenbaum, 1978; Michalski, 1977) evolved into the field of classification rule discovery (Clark and Niblett, 1989; Clearwater and Provost, 1990; Cohen, 1995; Segal and Etzioni, 1994; Webb, 1995). Work in affinity analysis led to the development of association rule discovery techniques (Agrawal et al., 1993; Agrawal and Srikant, 1994; Agrawal et al., 1996). Classification rule discovery is characterized by a concern for finding highly predictive rules, often using heuristic techniques. Association rule discovery is characterized by complete search to discover all rules that satisfy minimum bounds on support (the frequency with which the antecedent and consequent jointly occur) and other metrics such as confidence.

Techniques developed in the classification rule discovery paradigm are applicable when the primary objective is to obtain a classification capability. They are less useful, however, where the primary objective is to increase understanding of a domain by identifying unexpected regularities. While classification learning techniques will identify regularities, they will only identify one type of regularity, antecedents that are highly predictive of a consequent. To use association rule terminology, they are restricted to finding rules with very high confidence. However, as is well understood in the association rule community, predictiveness is not a universal measure of how interesting a rule is likely to be (Piatetsky-Shapiro, 1991). The interestingness of a rule will often be a function of both the strength of the correlation between the antecedent and the consequent and the expected strength of that correlation if the two were independent. For example, a rule buys lingerie $\rightarrow$ buys confectionary with confidence 0.95 (meaning 95% of customers that buy lingerie also buy confectionary) is unlikely to be of interest if 95% of all customers buy confectionary.

Another deficiency of classification rule discovery techniques when applied to gain insight (as opposed to gain classification capability), is that most classification rule discovery techniques seek to find a small set of rules with high coverage. In this context, the antecedents of the rules that are identified will not usually substantially overlap. In consequence, where there are alternative rules that apply to a particular class of records (for example rules that have either pregnant or female in their antecedents) only one is likely to be generated and presented to the user. Often, however, the relative interestingness of these rules will depend upon factors that it is infeasible to capture and represent to the data mining system. Hence it is valuable to have both alternatives presented to the user, who is then empowered to select which is of greater interest.

Association rules have a different set of strengths and limitations. They allow the use of interestingness metrics that consider the difference between the observed and expected degree of correlation and find all rules that satisfy user defined constraints. However, they are based on the application of a minimum support constraint. This constraint is required in order to prune the search space and make computation feasible. However, support is often not directly related to the interestingness of the rule. One example of this is the so called vodka and caviar problem (Cohen et al., 2000). A strong correlation between Ketel vodka and Beluga caviar may be of considerable interest (as they are high profit items and hence the affinity is likely to have high financial implications), even though they are very low volume items that will have low support. Where support is not directly related to the interestingness of a rule, the application of a minimum support constraint carries a risk that the most interesting rules will not be discovered.

In this paper we provide a general characterization of the rule discovery task that encompasses both classification and association rule discovery as well as supporting new categories of rule discovery activity. We provide an algorithm that supports rule discovery for a large and useful subset of these generalized rule discovery tasks and argue that there is a valuable role for the new types of rule discovery that this algorithm supports.

2. Association and Classification Rule Discovery

In this section we first provide a high level description of association rule discovery and then outline some key characteristics that distinguish classification rule discovery.

2.1. Association Rule Discovery

The association rule discovery task can be characterized as follows.

- A dataset $D$ is a finite set of records, where each record is an element to which we apply Boolean predicates called conditions.

- An itemset $I$ is a set of conditions. The name itemset derives from association rule discovery’s origins in market basket analysis where each condition denotes the presence of an item in a market basket.

- $\text{coverset}(I)$ denotes the set of records from a dataset that satisfy itemset $I$.

- An association rule consists of two conjunctions of conditions called the antecedent and consequent. An association rule with antecedent $a$ and consequent $c$ is denoted as $a \rightarrow c$.

- the support of an association rule $a \rightarrow c = |\text{coverset}(a \cup c)|/|D|$.

- the confidence of an association rule $a \rightarrow c = |\text{coverset}(a \cup c)|/|\text{coverset}(a)|$.

The task involves finding all association rules that satisfy user defined constraints on minimum support and confidence with respect to a given dataset.

Most association rule discovery algorithms utilize the frequent itemset strategy as exemplified by the Apriori algorithm (Agrawal et al., 1993). The frequent itemset strategy first discovers all frequent itemsets $\{I : |\text{coverset}(I)|/|D| \geq \text{min\_support}\}$, those itemsets whose support exceeds a user defined threshold $\text{min\_support}$. Association rules are then generated from these frequent itemsets. This approach is very efficient if there are relatively few frequent itemsets. It is, however, subject to a number of limitations.

1. There may be no natural lower bound on support. Associations with support lower than the nominated $\text{min\_support}$ will not be discovered. Infrequent itemsets may actually be especially interesting for some applications. As illustrated by the vodka and caviar problem, in many applications high value transactions are likely to be both relatively infrequent and of great interest.
2. Even if there is a natural lower bound on support the analyst may not be able to identify it. If \( \text{min\_support} \) is set too high then important associations will be overlooked. If it is set too low then processing may become infeasible. There is no means of determining, even after an analysis has been completed, whether it may have overlooked important associations due to the lower bound on support being set too high.

3. Even when a relevant minimum frequency can be specified, the number of frequent itemsets may be too large for computation to be feasible. Many datasets are infeasible to process using the frequent itemset approach with sensible specifications of minimum support (Bayardo, 1998).

4. The frequent itemset approach does not readily admit to techniques for improving efficiency by using constraints on the properties of rules that cannot be derived directly from the properties only of the antecedent, the consequent, or the union of the antecedent and consequent. Thus, they can readily benefit from a constraint on support (which depends solely on the frequency of the union of the antecedent and consequent) but cannot readily benefit from a constraint on confidence (which relates to the relationship between the support of the antecedent and of the union of the antecedent and the consequent). Where such constraints can be specified, potential efficiencies are lost.

An extension of the frequent itemset approach allows \( \text{min\_support} \) to vary depending upon the items that an itemset contains (Liu et al., 1999). While this introduces greater flexibility to the frequent itemset strategy, it does not resolve any of the four issues identified above.

Most research in association rule discovery has sought to improve the efficiency of the frequent itemset discovery process (Agarwal et al., 2000; Han et al., 2000; Savasere et al., 1995; Toivonen, 1996, for example). This has not addressed any of the above problems, except the closed itemset approaches (Pasquier et al., 1999; Pei et al., 2000; Zaki, 2000), which reduce the number of itemsets required, alleviating the problems of point 3, but not addressing 1, 2 or 4.

2.2. Classification Rule Discovery

Whereas the primary objective of association rule discovery is to find rules that are interesting to the user, the primary objective of classification rule discovery is to find rules that can be used to accurately classify objects. Our work falls within the former category. We wish to find interesting rules. However, to do this we extend techniques developed in the classification rule discovery paradigm.

Classification rule discovery usually employs some variant of the covering algorithm (Clark and Niblett, 1989; Cohen, 1995; Michalski, 1977) to find successive rules of the form \( a \rightarrow c \), where, in common with association rules, \( a \) is a set of conditions, but, unlike association rules, the consequent \( c \) is a single condition, a specification that a record belongs to one of a set of disjoint class values. The covering algorithm repeatedly performs a search to discover a rule that optimizes an objective function with respect to the training examples that have not been covered by a rule discovered so far.

A classification rule usually consists of a conjunction of conditions called the antecedent and a single condition called the consequent. In keeping with the notation for association rule discovery, we use \( a \rightarrow c \) to denote a classification rule with antecedent \( a \) and consequent \( c \).

The classification rule search task involves finding a single classification rule that optimizes a user defined objective function with respect to a given dataset. Most classification rule discovery algorithms perform search through the space of combinations of conditions that may appear in the antecedent seeking the combination that optimize the objective function with respect to a given consequent.

3. Generalized rule discovery

A number of research groups have explored rule discovery paradigms that combine elements of association and classification rule discovery (Bayardo et al., 2000; Bayardo and Agrawal, 1999; Clearwater and Provost, 1990; Rymon, 1992). From classification rule discovery they retain the objectives of searching for rules that optimize an objective function and that restrict the consequent to a single
condition that represents a value of a single class variable. From association rule discovery they take the objective of finding multiple rules, abandoning the covering objective, seeking instead many rules that each individually optimize some objective function. These approaches have advantages over standard classification and association rule discovery in numerous applications. Unlike standard classification rule discovery algorithms they return all rules that satisfy the objective function. This allows the user to select between these rules on criteria that may be difficult to specify and quantify in a manner suitable for application by the rule discovery system. Unlike standard association rule discovery algorithms they do not have to apply a constraint on minimum support. This greatly increases their versatility and allows the application solely of objective functions that are directly applicable to a particular rule discovery task.

However, these new rule discovery algorithms restrict discovery to rules with a value of a prespecified class variable as the consequent, a restriction that limits their applicability in “classical” data mining contexts where users wish to perform a wide ranging exploration for unexpected patterns in data. Having to specify a class variable greatly limits the rules that might be found. It might be thought that the process could simply be repeated for each potential consequent. However, in many data mining applications there are many thousands of potential consequents and such a strategy would be infeasible.

We propose a new rule discovery paradigm, generalized rule discovery, that combines elements of classification and association rule discovery as in previous research, but without the requirement that a class variable be specified. The result is a very general and flexible approach to rule discovery that can support search for rules that satisfy a wider range of constraints than the minimum support and confidence constraints of association rule discovery while freeing the discovery task from the limitations of a prespecified class variable. We maintain that this new paradigm facilitates valuable new forms of rule discovery.

Note that this differs from approaches to classification rule discovery that use association rule discovery techniques to find rules for classification (Liu et al., 1998). Rather, we extend algorithms developed in the classification context to support the same objective as association rule discovery. Our new algorithm is aimed at discovering interesting rules rather than rules for classification.

3.1. Formal Description of Generalized Rule Discovery

A formal definition of the generalized rule discovery task is given in the following.

Definition 1. A generalized rule discovery task (abbreviated as GRD task) is a 4-tuple \( \langle A, C, D, M \rangle \) and the solution to the generalized rule discovery task is a set of rules each of which takes the form of \( X \rightarrow Y \), where

- \( A \): is a nonempty set of conditions, called antecedent conditions;
- \( C \): is a nonempty set of conditions, called consequent conditions;
- \( D \): is a nonempty set of records, called the dataset, where for each record \( d \in D \), \( \text{conditions}(d) \subseteq A \cup C \), where \( \text{conditions}(d) \) is the set of conditions that apply to \( d \). For any set of conditions \( S \subseteq A \cup C \), let \( \text{coverset}(S) = \{ d | d \in D \land S \subseteq \text{conditions}(d) \} \), and let \( \text{cover}(S) = \frac{|\text{coverset}(S)|}{|D|} \);
- \( M \): is a set of constraints on the rules that form the solution for the generalized rule discovery task;
- \( X \): is a nonempty set of conditions, called the antecedent;
- \( Y \): is a nonempty set of conditions, called the consequent;

solution : \( \langle A, C, D, M \rangle \rightarrow \{ X \rightarrow Y \} \) is a many-to-one function mapping a GRD task to its solution, satisfying \( \text{solution}(\langle A, C, D, M \rangle) = \{ X \rightarrow Y | X \subseteq A \land Y \subseteq C \land X \rightarrow Y \text{ satisfies all constraints in } M \text{ with respect to } D \} \).
Generalized rule discovery separates the available conditions into two sets, the antecedent conditions and the consequent conditions. These specify, respectively, the conditions that may appear in the antecedent or the consequent of a solution to a GRD task. These two sets may overlap. These are distinguished in the specification of a GRD task in recognition that in many data mining contexts some conditions represent factors that may be directly manipulated while others represent outcomes that users would like to influence by manipulation of the first set of conditions. In such a context the user will be interested in rules where the antecedent is a selection from the first set of conditions and the consequent is a selection from the second set. This is because such rules can be operationalized. For example, conditions that represent identifiable characteristics of a customer make natural antecedents because they can be manipulated by strategies to acquire customers with specific profiles or by selecting such customers for a specific action once acquired. Conditions that represent a customer’s propensity to engage in a particular type of action make natural consequents, as it is for their propensity to act that we want to select a particular group.

3.2. Configuration

To emulate association rule discovery, the antecedent conditions and consequent conditions should both be the set of all conditions that appear in the dataset, and the set of constraints $\mathcal{M}$ should consist of constraints on the minimum allowed values for support and confidence. To emulate search for rules within classification rule discovery, the consequent conditions should be set to the values of the class variable, the antecedent conditions should be set to all other conditions, and $\mathcal{M}$ should constrain the solution to the rule that optimizes the objective function.

In the current work, however, we utilize a set of constraints that illustrate the manner in which GRD can support types of rule discovery not supported by other existing rule discovery paradigms. To this end we define four measures with respect to a rule $X \rightarrow Y$:

\[
\text{coverage}(X \rightarrow Y) = \text{cover}(X),
\]
\[
\text{support}(X \rightarrow Y) = \text{cover}(X \cup Y),
\]
\[
\text{confidence}(X \rightarrow Y) = \frac{\text{support}(X \rightarrow Y)}{\text{coverage}(X \rightarrow Y)},
\]
\[
\text{leverage}(X \rightarrow Y) = \text{support}(X \rightarrow Y) - \text{cover}(X) \times \text{cover}(Y).
\]

Piatetsky-Shapiro (1991) argues that many measures of interestingness are based on the difference between the observed joint frequency of the antecedent and consequent, $\text{support}(X \rightarrow Y)$, and the frequency that would be expected if the two were independent, $\text{cover}(X) \times \text{cover}(Y)$. He asserts that the simplest such measure is one that we call leverage, as defined above. Note that leverage can also be expressed as $\text{cover}(X) \times (\text{confidence}(X \rightarrow Y) - \text{cover}(Y))$. Expressed in this form, it has also been called weighted relative accuracy (Todorovski et al., 2000).

Leverage is of interest because it measures the number of additional records that an interaction involves above and beyond those that should be expected if one assumes independence. This directly represents the volume of an effect and hence will often directly relate to the ultimate metric of interest to the user such as the magnitude of the profit associated with the interaction between the antecedent and consequent. This contrasts with the traditional association rule measure

\[
\text{lift}(X \rightarrow Y) = \frac{\text{support}(X \rightarrow Y)}{\text{cover}(X) \times \text{cover}(Y)}
\]

which is the ratio of the observed frequency with which the consequent occurs in the context of the antecedent over that expected if the two were independent. A rule with high lift may be of little interest because it applies very infrequently. This might be provided as justification for the application of minimum support constraints in conjunction with the lift measure, but this is at best a crude approximate fix to the problem. It results in a step function with the very undesirable property that the addition of one more record in support of a rule with very high lift can transform it from
being of no interest whatsoever because it fails to meet a minimum support criterion to being of very high interest because it now meets that criterion. In contrast to the use of support and lift as measures of interest, the use of leverage ensures both that rules with very low support will receive low values and that there are no artificial threshold values at which dramatic changes in the objective function occur.

With these considerations in mind, for the purposes of this paper we restrict the set of constraints $\mathcal{M}$ to constraints on the coverage, support, confidence, and leverage of a rule. $\mathcal{M}$ is composed of

$maxRules$ denoting the maximum number of rules in the solution (which will consist of the rules with the highest values for leverage of those that satisfy all other constraints),

$maxLHSsize$ denoting maximum number of conditions allowed on the antecedent of rule

$minCoverage$ denoting the minimum coverage,

$minSupport$ denoting the minimum support,

$minConfidence$ denoting the minimum confidence, and

$minLeverage = \max(0.0, \beta(RS, maxRules))$, where $RS$ is the set of rules $\{R|coverage(R) \geq minCoverage \land support(R) \geq minSupport \land confidence(R) \geq minConfidence\}$, and $\beta(Z,n)$ is the leverage of the $n^{th}$ rule in $Z$ sorted from highest to lowest by leverage.

These user configured constraints are augmented by the constraints that the antecedent of a rule is composed of at least one condition and the consequent of rule is composed of single condition.

In consequence, in the current work, $solution : (A, C, D, \mathcal{M}) \rightarrow \{X \rightarrow Y\}$ is a many-to-one function mapping a $GRDtask$ to its solution, satisfying $solution((A, C, D, \mathcal{M})) = \{X \rightarrow Y | X \subseteq A \land Y \subseteq C \land 1 \leq |X| \leq maxLHSsize \land |Y| = 1 \land coverage(X \rightarrow Y) \geq minCoverage \land support(X \rightarrow Y) \geq minSuport \land confidence(X \rightarrow Y) \geq minConfidence \land leverage(X \rightarrow Y) \geq minLeverage\}$.

## 4. The OPUS Search Algorithm

We propose an algorithm for solving a useful class of GRD tasks based on the OPUS (Webb, 1995) search algorithm. OPUS provides efficient search for subset selection, such as selecting a subset of available conditions that optimizes a specified measure. It was developed for classification rule discovery. Prior to the development of OPUS, classification rule discovery algorithms that performed complete search ordered the available conditions and then conducted a systematic search over the ordering in such a manner as to guarantee that each subset was investigated once only, as illustrated in Figure 1.

Critical to the efficiency of such search is the ability to identify and prune sections of the search space that cannot contain solutions to the search task. This is usually achieved by identifying subsets that cannot appear in a solution. For example, it might be determined that no superset of $\{b\}$ can be
a solution in the search space illustrated in Figure 1. Under previous search algorithms (Clearwater and Provost, 1990; Morishita and Nakaya, 2000; Provost et al., 1999; Rymon, 1992; Segal and Etzioni, 1994), the search space below such a node was pruned, as illustrated in Figure 2. In this example, pruning removes one subset from the search space.

This contrasts with the pruning that would occur if all supersets of the pruned subset were removed from the search space, as illustrated in Figure 3. This optimal pruning almost halves the search space below the parent node.

OPUS achieves the pruning illustrated in Figure 3 by maintaining a set of available items at each node in the search space. When adding an item \(i\) to the current subset \(s\) results in a subset \(s \cup \{i\}\) that can be pruned from the search space, \(i\) is simply removed from the set of available items at \(s\) which is propagated below \(s\). As supersets of \(s \cup \{i\}\) below \(s\) can only be explored after \(s \cup \{i\}\), this simple mechanism with negligible computational overheads guarantees that no superset of a pruned subset will be generated in the search space below the parent of the pruned node. This greatly expands the scope of a pruning operation from that achieved by previous algorithms which only extended to the space directly beneath the pruned node. Further pruning can be achieved by reordering the search space (Webb, 1995). However, this proves to be infeasible in generalized rule discovery search as there is a large amount of information associated with each node in the search space (specifically, the set of records covered by the current antecedent) and it is more efficient to utilize this information when it is first calculated than to either store it for later use or recalculate it at a later stage as would be required if the nodes were reordered before expansion.

Note that nodes are pruned only when they need not be explored during the search. Nodes may need to be explored even when they are not candidate solutions because candidate solutions might occur deeper in the search space below those nodes. Pruning is applied only when this is not the case. In consequence, OPUS does not require either monotonicity or anti-monotonicity from its objective function. It requires only that the value of the objective function can be bounded so that branch and bound techniques can exclude sections of the search space from exploration.

It is interesting to observe that while the rule discovery systems Max-Miner and Dense-Miner have been described as using SE-Tree search (Bayardo, 1998; Bayardo et al., 2000), they are perhaps more
accurately described as using OPUS search, as they use both the propagation of a set of available items and search space reordering, two strategies that distinguish OPUS search from SE-Tree search.

5. The GRD Algorithm

The GRD algorithm extends the OPUS search algorithm to generalized rule discovery (Webb, 2000). To simplify the search problem, the consequent of a rule is restricted to a single condition. Rules of this restricted form are of interest for many data mining applications, and indeed many association rule discovery systems also impose such a constraint.

Whereas OPUS supports search through spaces of subsets, the generalized rule discovery task requires search through the space of pairs \( (a \subseteq \text{AntecedentConditions}, c \in \text{ConsequentConditions}) \), where \( a \) is the antecedent and \( c \) the consequent of rule. GRD achieves this by performing OPUS search through the space of antecedents, maintaining at each node a set of potential consequents, each of which is explored at each node.

GRD extends previous algorithms that perform OPUS-like search for rule discovery (Bayardo, 1998; Bayardo et al., 2000) by removing the requirement that the consequent of each of the rules discovered be one of the values of a single ‘class’ variable. Unlike Bayardo and Agrawal’s (1999) technique that finds all rules that can maximize any of a number of widely used measures of interestingness, GRD finds all rules that satisfy a specified set of constraints. These may include rules that achieve high scores but not the highest according to the primary measure of interestingness. We believe that this is important, as it is often not possible for the user to provide formal measures of all criteria that relate to the degree of interestingness of a rule. It can be valuable for the user to be provided with a range of alternative rules that score highly on the specified interestingness measure, in order that they can then apply further criteria to select the rules of greatest interest, such as the feasibility of operationalizing business plans based on a rule.

The algorithm relies upon there being a set of user defined constraints on the acceptable rules. These are used to prune the search space. Such constraints can take many forms, ranging from the traditional association rule discovery constraints on support and confidence to a constraint that only the \( n \) rules that maximize some statistic be returned. The solution to a GRD task is the set of rules that satisfy all the constraints. Note that it may not be apparent when a rule is encountered whether or not it is in the solution. For example, if we are seeking the 100 rules with the highest leverage, we may not know the cutoff value for leverage until the search has been completed.

To manage this problem, GRD is restricted to GRD tasks \( (\mathcal{A}, \mathcal{C}, \mathcal{D}, \mathcal{M}) \) that satisfy the following property:

\[
\forall R \subseteq \{ W \rightarrow Z \mid W \subseteq \mathcal{A} \land Z \subseteq \mathcal{C}\}, \quad \text{solution}(\langle \mathcal{A}, \mathcal{C}, \mathcal{D}, \mathcal{M}\rangle) \cap R \subseteq \text{solution}(\langle \mathcal{A}, \mathcal{C}, \mathcal{D}, \mathcal{M} \land X \rightarrow Y \in R \rangle) \quad (1)
\]

This allows an incremental search to be performed. In this search, rules in the search space are considered one at a time. Let \( S \) represent the set of rules in the search space explored so far. A record currentSolution is maintained such that currentSolution = solution(\( (\mathcal{A}, \mathcal{C}, \mathcal{D}, \mathcal{M} \land X \rightarrow Y \in S) \)). For each new rule \( r \) considered during the search, the algorithm need only update currentSolution by

- adding \( r \) if \( r \in \text{solution}(\langle \mathcal{A}, \mathcal{C}, \mathcal{D}, \mathcal{M} \land X \rightarrow Y \in S \cup \{r\}\rangle) \), and
- removing any \( z \in \text{currentSolution} \) if \( z \notin \text{solution}(\langle \mathcal{A}, \mathcal{C}, \mathcal{D}, \mathcal{M} \land X \rightarrow Y \in S \cup \{r\}\rangle) \).

Appendix A presents a proof that the update strategy for currentSolution is sufficient and necessary to ensure that on termination currentSolution = solution(\( (\mathcal{A}, \mathcal{C}, \mathcal{D}, \mathcal{M}) \)).

This simple constraint, (1), greatly simplifies the search task. The constraint is not unduly restrictive as it is satisfied by all \( \mathcal{M} \) that we have considered to date. Note in particular that it allows monotone and anti-monotone constraints, as well as constraints that are neither. Our algorithm uses branch-and-bound techniques, and hence relies on neither monotonicity nor anti-monotonicity in its
constraints. The objective function, leverage, used in our experiments is non-monotone. The addition of a new condition to an antecedent can raise, lower or leave unaffected the leverage of a rule.

Table I displays the algorithm that results from extending the OPUS search algorithm (Webb, 1995) to obtain efficient search for this rule discovery task. The algorithm is presented as a recursive procedure with three arguments:

**CurrentLHS:** the set of conditions in the antecedent of the rule currently being considered.

**AvailableLHS:** the set of conditions that may be added to the antecedent of rules to be explored below this point.

**AvailableRHS:** the set of conditions that may appear on the consequent of a rule in the search space at this point and below.

The algorithm also maintains a global variable `currentSolution`, the solution to GRD task constrained to the rules in the search space considered so far.

To solve GRD task \( \langle A, C, D, M \rangle \), `currentSolution` is initialized to \( \emptyset \) and the initial call to GRD is made with `CurrentLHS`=\( \emptyset \), `AvailableLHS`=\( A \), and `AvailableRHS`=\( C \).

The GRD algorithm is a search procedure that starts with rules with one condition in the antecedent and searches through successive rules formed by adding conditions to the antecedent. It loops through each condition in `AvailableLHS` and adds it to `CurrentLHS` to form the `NewLHS`. For the `NewLHS`, it loops through each condition \( c \) in `AvailableRHS` to check whether `NewLHS` \( \rightarrow c \) might be in the solution. After the `AvailableRHS` loop, the procedure is recursively called with the arguments `NewLHS`, `NewAvailableLHS` and `NewAvailableRHS`. The two latter arguments are formed by removing the pruned conditions from `AvailableLHS` and `AvailableRHS`, respectively.

Pruning rules seek to identify areas of the search space that cannot contain a solution. This is represented within the algorithm by the use of a predicate `insolution(a \rightarrow c, T)` that is true iff the rule \( a \rightarrow c \) is in the solution to the GRD task, \( T \). The predicate `proven(X)` is true iff pruning rules provided to the algorithm prove the proposition \( X \). The use of this predicate allows us to abstract the algorithm from the sets of pruning rules that might be used to provide efficient search for a given set of constraints. The efficiency of GRD will depend critically on the power of the pruning rules with which it is provided.

As GRD traverses the space of rules, the `minLeverage` constraint is increased dynamically so that it is always the `maxRules`\(^{th} \)` leverage value of all the rules searched so far. Line 19 records the rules satisfying the current constraints. Whenever adding a rule to `currentSolution` causes the number of rules in `currentSolution` to exceed `maxRules`, Line 20 removes the unqualified rule whose leverage value ranks (`maxRules + 1`). When the search is finished, `currentSolution` is the solution to the GRD task.

Appendix B proves the correctness of the GRD\(^{\prime} \)` algorithm, a variant of GRD with all pruning removed. In the following section, we will list the pruning techniques adopted in GRD. Their correctness lies in the theorems on properties of rules in the solution and relations between rules. As GRD\(^{\prime} \)` is correct, GRD will also be correct so long as all the pruning rules are correct.

### 6. Pruning in Search for Rules

#### 6.1. Properties of the Rules in the Solution of GRD Task

To facilitate the analysis of the GRD algorithm, we analyze some necessary properties of GRD tasks and their solutions. Before presenting the theorems, we give the following lemma.

**Lemma 1 (Subset cover).** Suppose GRD task \( \langle A, C, D, M \rangle \). For any \( S_1, S_2 \subseteq A \cup C \), if \( S_1 \subseteq S_2 \), then `coverset(S_2)` \( \subseteq `coverset(S_1)``, and hence, `cover(S_2)` \( \leq `cover(S_1)`` holds.

**Proof.** For any \( d \in `coverset(S_2)``, according to the definition of GRD task, \( S_2 \subseteq `conditions(d)` \) holds. Since \( S_1 \subseteq S_2 \), \( S_1 \subseteq `conditions(d)` \) holds. Hence \( d \in `coverset(S_1)``. So `coverset(S_2)` \( \subseteq `coverset(S_1)` \) holds.

\( \square \)
Table I. The generalized rule discovery algorithm

Algorithm: GRD(CurrentLHS,AvailableLHS,AvailableRHS)
1: SoFar := ∅
2: for all P in AvailableLHS do
3: if ¬proven(∀x ⊆ AvailableLHS: ∀y ∈ AvailableRHS: ¬insolution(x ∪ CurrentLHS ∪ {P} → y, ⟨A,C,D,M⟩)) then
4:   NewLHS := CurrentLHS ∪ {P}
5:   NewAvailableLHS := SoFar
6:   if ¬proven(∀x ⊆ NewAvailableLHS: ∀y ∈ AvailableRHS: ¬insolution(x ∪ NewLHS → y, ⟨A,C,D,M⟩)) then
7:     if P in AvailableRHS then
8:       NewAvailableRHS := AvailableRHS - P
9:     else
10:    NewAvailableRHS := AvailableRHS
11:  end if
12:  if ¬proven(∀y ∈ NewAvailableRHS: ¬insolution(NewLHS → y, ⟨A,C,D,M⟩)) then
13:    for all Q in NewAvailableRHS do
14:      if proven(∀x ⊆ NewAvailableLHS: ¬insolution(x ∪ NewLHS → Q, ⟨A,C,D,M⟩)) then
16:      else
17:        if ¬proven(¬insolution(NewLHS → Q, ⟨A,C,D,M⟩)) then
18:          if insolution(NewLHS → Q, ⟨A,C,D,M ∧ X → Y ∈ currentSolution ∪ {NewLHS → Q})
19:            add NewLHS → Q to currentSolution
20:            remove from currentSolution any rule W → Z: ¬insolution(W → Z, ⟨A,C,D,M ∧ X → Y ∈ currentSolution ∪ {NewLHS → Q})
21:            tune the settings of the constraints
22:          end if
23:        end if
24:    end if
25: end if
26: end if
27: end for
28: end if
29: end if
30: end if
31: if NewAvailableLHS ≠ ∅ and NewAvailableRHS ≠ ∅ then
32:   GRD(NewLHS,NewAvailableLHS,NewAvailableRHS)
33: end if
34: end if
35: SoFar := SoFar ∪ {P}
36: end for

Theorem 1 (Minimum coverage for rules in the solution). Suppose GRDtask = ⟨A,C,D,M⟩. For any rule X → Y, if there exists X₁ ⊆ X satisfying cover(X₁) < minCoverage, X → Y ∉ solution(⟨A,C,D,M⟩).

Proof. According to the definition of GRDtask, X₁ ⊆ X and the subset cover lemma, the following holds.

\[
\text{coverage}(X → Y) = \text{cover}(X) \leq \text{cover}(X₁) < \text{minCoverage}
\]

Hence X → Y ∉ solution(⟨A,C,D,M⟩).

Theorem 2 (Minimum support for rules in the solution). Suppose GRDtask = ⟨A,C,D,M⟩. For any rule X → Y, if there exists Z ⊆ X ∪ Y satisfying cover(Z) < minSupport, X → Y ∉ solution(⟨A,C,D,M⟩).
Proof. According to the definition of \textit{GRDtask}, $Z \subseteq X \cup Y$ and the subset cover lemma, the following holds.

$$\text{support}(X \rightarrow Y) = \text{cover}(X \cup Y) \leq \text{cover}(Z) < \text{minSupport}$$

Hence $X \rightarrow Y \notin \text{solution}(\langle A, C, D, M \rangle)$.

\textbf{Theorem 3 (Minimum leverage for rules in the solution).} Suppose \textit{GRDtask} = $\langle A, C, D, M \rangle$. For any rule $X \rightarrow Y$, if $\text{cover}(Y) > 1 - \frac{\text{minLeverage}}{\text{cover}(X)}$ or $\text{cover}(X) > 1 - \frac{\text{minLeverage}}{\text{cover}(Y)}$, $X \rightarrow Y \notin \text{solution}(\langle A, C, D, M \rangle)$.

\textit{Proof.} Let us first prove the theorem when $\text{cover}(Y) > 1 - \frac{\text{minLeverage}}{\text{cover}(X)}$. According to the definition of \textit{GRDtask}, we obtain:

$$\text{leverage}(X \rightarrow Y) = \text{support}(X \rightarrow Y) - \text{cover}(X) \times \text{cover}(Y) = \text{cover}(X \cup Y) - \text{cover}(X) \times \text{cover}(Y)$$

From $\text{cover}(Y) > 1 - \frac{\text{minLeverage}}{\text{cover}(X)}$ we obtain $\text{cover}(X) \times \text{cover}(Y) > \text{cover}(X) - \text{minLeverage}$, thus $\text{leverage}(X \rightarrow Y)$ satisfies:

$$\text{leverage}(X \rightarrow Y) < \text{cover}(X \cup Y) - (\text{cover}(X) - \text{minLeverage}) = \text{cover}(X \cup Y) - \text{cover}(X) + \text{minLeverage}$$

From the subset cover lemma we obtain $\text{cover}(X \cup Y) \leq \text{cover}(X)$. Thus the following holds.

$$\text{leverage}(X \rightarrow Y) < \text{minLeverage}$$

Hence $X \rightarrow Y \notin \text{solution}(\langle A, C, D, M \rangle)$. Similarly the theorem is proved when $\text{cover}(X) > 1 - \frac{\text{minLeverage}}{\text{cover}(X)}$.

\textbf{Theorem 4 (Minimum leverage for the antecedent of rules in the solution).} Suppose \textit{GRDtask} = $\langle A, C, D, M \rangle$. For any rule $X \rightarrow Y$, if $\text{cover}(X) \times (1 - \text{cover}(X)) < \text{minLeverage}$, $X \rightarrow Y \notin \text{solution}(\langle A, C, D, M \rangle)$.

\textit{Proof.} Firstly let us assume $\text{cover}(X) \leq \text{cover}(Y)$. Thus $1 - \text{cover}(X) \geq 1 - \text{cover}(Y)$ holds. According to the definition of \textit{GRDtask} and the subset cover lemma, we obtain:

$$\text{leverage}(X \rightarrow Y) = \text{support}(X \rightarrow Y) - \text{cover}(X) \times \text{cover}(Y) = \text{cover}(X \cup Y) - \text{cover}(X) \times \text{cover}(Y)$$

$$\leq \text{cover}(X) - \text{cover}(X) \times \text{cover}(Y) = \text{cover}(X) \times (1 - \text{cover}(Y))$$

$$\leq \text{cover}(X) \times (1 - \text{cover}(X)) < \text{minLeverage}$$

Secondly let us assume $\text{cover}(X) > \text{cover}(Y)$. Thus $\frac{\text{cover}(Y)}{\text{cover}(X)} < 1$ holds. According to the definition of \textit{GRDtask} and the subset cover lemma, we obtain:

$$\text{leverage}(X \rightarrow Y) = \text{support}(X \rightarrow Y) - \text{cover}(X) \times \text{cover}(Y) = \text{cover}(X \cup Y) - \text{cover}(X) \times \text{cover}(Y)$$

$$\leq \text{cover}(Y) - \text{cover}(X) \times \text{cover}(Y) = \text{cover}(Y) \times (1 - \text{cover}(X))$$

$$< \text{cover}(Y) \times \frac{\text{minLeverage}}{\text{cover}(X)} < \text{minLeverage}$$

So for both cases $\text{leverage}(X \rightarrow Y) < \text{minLeverage}$ holds. Hence $X \rightarrow Y \notin \text{solution}(\langle A, C, D, M \rangle)$.

\textbf{Theorem 5 (Minimum confidence for rules in the solution).} Suppose \textit{GRDtask} = $\langle A, C, D, M \rangle$. For any rule $X \rightarrow Y$, if $\frac{\text{cover}(Y)}{\text{cover}(X)} < \text{minConfidence}$, $X \rightarrow Y \notin \text{solution}(\langle A, C, D, M \rangle)$.

\textit{Proof.} According to the definition of \textit{GRDtask} and the subset cover lemma, we obtain:

$$\text{confidence}(X \rightarrow Y) = \frac{\text{support}(X \rightarrow Y)}{\text{cover}(X \rightarrow Y)} = \frac{\text{cover}(X \cup Y)}{\text{cover}(X)} \leq \frac{\text{cover}(Y)}{\text{cover}(X)}$$

From $\frac{\text{cover}(Y)}{\text{cover}(X)} < \text{minConfidence}$, $\text{confidence}(X \rightarrow Y) < \text{minConfidence}$ holds. Hence, $X \rightarrow Y \notin \text{solution}(\langle A, C, D, M \rangle)$. 

\textbf{GRD.tex; 21/03/2003; 13:31; p.11}
Theorem 6 (Full cover). Suppose $GRDtask = \langle A, C, D, M \rangle$. For any rule $X \to Y$, if $cover(X) = 1$ or $cover(Y) = 1$, $leverage(X \to Y) = 0$ holds.

Proof. When $cover(X) = 1$, $cover(X \cup Y) = cover(Y)$ holds. Thus the following holds.

$leverage(X \to Y) = support(X \to Y) - cover(X) \times cover(Y) = cover(Y) - cover(Y) = 0$

Similarly the theorem is proved when $cover(Y) = 1$. $\square$

Theorem 7 (Coverage for rules in the solution). Suppose $GRDtask = \langle A, C, D, M \rangle$. For any rule $X \to Y \in solution(\langle A, C, D, M \rangle)$, $coverage(X \to Y) \geq minSupport$ holds.

Proof. According to the definition of $GRDtask$, the subset cover lemma and that $X \to Y$ is in the solution, we obtain:

$coverage(X \to Y) = cover(X) \geq cover(X \cup Y) = support(X \to Y) \geq minSupport$

$\square$

Theorem 8 (Support for rules in the solution). Suppose $GRDtask = \langle A, C, D, M \rangle$. For any rule $X \to Y \in solution(\langle A, C, D, M \rangle)$, $support(X \to Y) \geq minCoverage \times minConfidence$ holds.

Proof. According to the definition of $GRDtask$ and that $X \to Y$ is in the solution, we obtain:

$support(X \to Y) = coverage(X \to Y) \times confidence(X \to Y) \geq minCoverage \times minConfidence$

$\square$

Theorem 9 (Support for rules in the solution related to leverage). Suppose $GRDtask = \langle A, C, D, M \rangle$. For any rule $X \to Y \in solution(\langle A, C, D, M \rangle)$, $support(X \to Y) \geq minLeverage$ holds.

Proof. According to the definition of $GRDtask$ and that $X \to Y$ is in the solution, we obtain:

$support(X \to Y) = leverage(X \to Y) + cover(X) \times cover(Y) \geq leverage(X \to Y) \geq minLeverage$

$\square$

6.2. Relations between rules in GRDtask

We investigate the relations between two rules when they share the same coverage value under some condition. We start with the following lemma.

Lemma 2 (Union cover). Suppose $GRDtask = \langle A, C, D, M \rangle$. For any nonempty $S_1, S_2, S_3 \subseteq A \cup C$ satisfying $S_1 \cap S_2 = \emptyset$, $S_2 \cap S_3 = \emptyset$ and $S_1 \cap S_3 = \emptyset$, if

$cover(S_1) = cover(S_1 \cup S_2)$ (2)

the following holds.

$cover(S_1 \cup S_3) = cover(S_1 \cup S_2 \cup S_3)$ (3)

Proof. From (2) and the definition of $GRDtask$, we obtain:

$|coverset(S_1)| = |coverset(S_1 \cup S_2)|$ (4)

From the subset cover lemma, we obtain:

$coverset(S_1) \supseteq coverset(S_1 \cup S_2)$ (5)

From (4) and (5), we obtain:

$coverset(S_1) = coverset(S_1 \cup S_2)$ (6)
For any $d \in \mathcal{D} \land S_1 \cup S_3 \subseteq \text{conditions}(d)$, $S_1 \subseteq \text{conditions}(d)$ and $S_3 \subseteq \text{conditions}(d)$ hold. From $S_1 \subseteq \text{conditions}(d)$ and (6), we obtain $S_1 \cup S_2 \subseteq \text{conditions}(d)$. And since $S_3 \subseteq \text{conditions}(d)$, $S_1 \cup S_2 \cup S_3 \subseteq \text{conditions}(d)$ holds. Hence we obtain:

$$\text{coverset}(S_1 \cup S_3) \subseteq \text{coverset}(S_1 \cup S_2 \cup S_3)$$

(7)

From the subset cover lemma, we obtain:

$$\text{coverset}(S_1 \cup S_3) \supseteq \text{coverset}(S_1 \cup S_2 \cup S_3)$$

(8)

From (7) and (8), $\text{coverset}(S_1 \cup S_3) = \text{coverset}(S_1 \cup S_2 \cup S_3)$ holds, proving (3).

**Theorem 10 (Relation from confidence).** Suppose $\text{GRDtask} = \langle \mathcal{A}, \mathcal{C}, \mathcal{D}, \mathcal{M} \rangle$. For any rule $X \rightarrow Y$, if $\text{confidence}(X \rightarrow Y) = 1$, for any $X_1 \subseteq \mathcal{A}$ satisfying $X_1 \cap X = \emptyset$ and $X_1 \cap Y = \emptyset$, the following holds.

$$\text{leverage}(X \cup X_1 \rightarrow Y) \leq \text{leverage}(X \rightarrow Y)$$

**Proof.** From $\text{confidence}(X \rightarrow Y) = 1$ and the definition of $\text{GRDtask}$, we obtain:

$$\text{cover}(X) = \text{cover}(X \cup Y)$$

(9)

According to the definition of $\text{GRDtask}$ and (9), we obtain:

$$\text{leverage}(X \rightarrow Y) = \text{support}(X \rightarrow Y) - \text{cover}(X) \times \text{cover}(Y)$$

$$= \text{cover}(X \cup Y) - \text{cover}(X) \times \text{cover}(Y)$$

$$= \text{cover}(X) - \text{cover}(X) \times \text{cover}(Y)$$

(10)

From (9) and the union cover lemma, we obtain:

$$\text{cover}(X \cup X_1) = \text{cover}(X \cup X_1 \cup Y)$$

(11)

According to the definition of $\text{GRDtask}$ and (11), we obtain:

$$\text{leverage}(X \cup X_1 \rightarrow Y) = \text{support}(X \cup X_1 \rightarrow Y) - \text{cover}(X \cup X_1) \times \text{cover}(Y)$$

$$= \text{cover}(X \cup X_1 \cup Y) - \text{cover}(X \cup X_1) \times \text{cover}(Y)$$

$$= \text{cover}(X \cup X_1) - \text{cover}(X \cup X_1) \times \text{cover}(Y)$$

(12)

From the subset cover lemma we obtain $\text{cover}(X \cup X_1) \leq \text{cover}(X)$. Thus, from (10) and (12), the following holds.

$$\text{leverage}(X \cup X_1 \rightarrow Y) \leq \text{leverage}(X \rightarrow Y)$$

**Theorem 11 (Relation from coverage).** Suppose $\text{GRDtask} = \langle \mathcal{A}, \mathcal{C}, \mathcal{D}, \mathcal{M} \rangle$. For any rule $X \rightarrow Y$ and $X \cup X_1 \rightarrow Y$ where $X \cap X_1 = \emptyset$, if

$$\text{coverage}(X \rightarrow Y) = \text{coverage}(X \cup X_1 \rightarrow Y)$$

(13)

the following holds.

$$\text{support}(X \rightarrow Y) = \text{support}(X \cup X_1 \rightarrow Y)$$

(14)

$$\text{confidence}(X \rightarrow Y) = \text{confidence}(X \cup X_1 \rightarrow Y)$$

(15)

$$\text{leverage}(X \rightarrow Y) = \text{leverage}(X \cup X_1 \rightarrow Y)$$

(16)
Proof. According to (13) which is $\text{cover}(X) = \text{cover}(X \cup X_1)$, and from the union cover lemma, we obtain:

$$
\text{cover}(X \cup Y) = \text{cover}(X \cup X_1 \cup Y)
$$

proving (14). From (13) and (14), obviously (15) and (16) are proved.

6.3. Pruning the condition before it is added to the antecedent

This pruning rule, applied at line 3 in the GRD algorithm, prunes a condition in $\text{AvailableLHS}$ before it is added to $\text{CurrentLHS}$.

**Pruning 1.** In GRD for $\text{GRDtask} = (A,C,D,M)$, for any condition $P \in \text{AvailableLHS}$, if $\text{cover}(\{P\}) < \text{minCoverage}$, $P$ can be pruned from $\text{SoFar}$.

According to the theorem of minimum coverage for rules in the solution, there does not exist any rule $X \rightarrow Y \in \text{solution}(\langle A,C,D,M \rangle)$ such that $P \in X$, thus $P$ can be pruned from $\text{SoFar}$ so that $P$ will not go into any $\text{NewAvailableLHS}$.

6.4. Pruning the new antecedent

This pruning rule, applied at line 6 in GRD, is used to prune the new antecedent $\text{NewLHS}$ which is formed by the union $\text{CurrentLHS} \cup \{P\}$ where $P \in \text{AvailableLHS}$.

**Pruning 2.** In GRD for $\text{GRDtask} = (A,C,D,M)$, for $\text{NewLHS} = \text{CurrentLHS} \cup \{P\}$ where $P \in \text{AvailableLHS}$, if $\text{cover}(\text{NewLHS}) < \text{minCoverage}$, $P$ can be pruned from $\text{SoFar}$.

According to the theorem of minimum coverage for rules in the solution, there does not exist any rule $X \rightarrow Y \in \text{solution}(\langle A,C,D,M \rangle)$ such that $\text{NewLHS} \subseteq X$, thus $P$ can be pruned from $\text{SoFar}$ so that $P$ will not go into any $\text{NewAvailableLHS}$ ready to be added to the antecedent containing $\text{CurrentLHS}$.

6.5. Pruning the consequent condition before the evaluation of rule

Pruning rules applied at line 14 in GRD are used to prune the consequent condition before the evaluation of a rule. We give three pruning rules.

**Pruning 3.** In GRD for $\text{GRDtask} = (A,C,D,M)$, for any condition $Q \in \text{NewAvailableRHS}$, if $\text{cover}(\{Q\}) < \text{minSupport}$, then $Q$ can be pruned from $\text{NewAvailableRHS}$.

According to the theorem of minimum support for rules in the solution, if $\text{cover}(\{Q\}) < \text{minSupport}$ then $\forall A \subseteq A, \text{support}(A \rightarrow Q) < \text{minSupport}$, therefore $Q$ can be pruned.

The second pruning rule functions according to the current lower bound on $\text{minLeverage}$ before the evaluation of the rule. Note that the lower bound on $\text{minLeverage}$ is the leverage of the $\text{maxRules}^{th}$ rule that satisfies the other criteria of those found so far, ordered from highest to lowest value on leverage.

**Pruning 4.** In GRD for $\text{GRDtask} = (A,C,D,M)$, for the current $\text{NewLHS}$, for any condition $Q \in \text{NewAvailableRHS}$, if $\text{cover}(\{Q\}) > 1 - \frac{\text{minLeverage}}{\text{cover}(\text{NewLHS})}$, then $Q$ can be pruned from $\text{NewAvailableRHS}$.

According to the theorem of minimum leverage for rules in the solution, if $\text{cover}(\{Q\}) > 1 - \frac{\text{minLeverage}}{\text{cover}(\text{NewLHS})}$ then $\text{NewLHS} \rightarrow Q$ cannot be in the solution, therefore $Q$ can be pruned.

The third pruning rule is for any consequent condition that covers the whole dataset.

**Pruning 5.** In GRD for $\text{GRDtask} = (A,C,D,M)$, for any condition $Q \in \text{NewAvailableRHS}$, if $\text{cover}(\{Q\}) = 1$ and $\text{minLeverage} > 0$, then $Q$ can be pruned from $\text{NewAvailableRHS}$.
According to the theorem of full cover, if \( \text{cover}(\{Q\}) = 1 \) then \( \forall A \subseteq \mathcal{A}, \text{leverage}(\text{NewLHS} \rightarrow Q) = 0 \). When \( \text{minLeverage} > 0 \), such a rule cannot be in the solution. Therefore \( Q \) can be pruned.

6.6. Pruning the consequent condition after the evaluation of rule

This pruning rule at line 23 in GRD is used to prune the consequent condition after the evaluation of the current rule. We give three pruning rules.

**Pruning 6.** In GRD for GRDtask = \( \langle \mathcal{A}, \mathcal{C}, \mathcal{D}, \mathcal{M} \rangle \), after the evaluation of the current rule \( \text{NewLHS} \rightarrow Q \), if \( \text{confidence}(\text{NewLHS} \rightarrow Q) = 1 \) and \( \text{leverage}(\text{NewLHS} \rightarrow Q) < \text{minLeverage} \), \( Q \) can be pruned from \( \text{NewAvailableRHS} \).

According to the theorem of relation from confidence, if \( \text{confidence}(\text{NewLHS} \rightarrow Q) = 1 \) then no rule with \( Q \) as the consequent in the search space below the current node can have higher leverage than \( \text{NewLHS} \rightarrow Q \). Therefore if \( \text{leverage}(\text{NewLHS} \rightarrow Q) < \text{minLeverage} \), none of these rules can be in the solution. Hence \( Q \) can be pruned from \( \text{NewAvailableRHS} \).

The second pruning rule utilizes \( \text{opt\_confidence} \), an upper bound on the value of the confidence for a rule with \( Q \) as the consequent in the search space below the current node.

**Pruning 7.** In GRD for GRDtask = \( \langle \mathcal{A}, \mathcal{C}, \mathcal{D}, \mathcal{M} \rangle \), after the evaluation of the current rule \( \text{NewLHS} \rightarrow Q \) where \( \text{NewLHS} = \text{CurrentLHS} \cup \{P\} \), \( P \in \text{AvailableLHS} \), \( \text{opt\_confidence} \) is computed by:

\[
\text{opt\_confidence} = \frac{\text{support}(\text{NewLHS} \rightarrow Q)}{\text{min\_cover}}
\]

where

\[
\text{min\_cover} = \max(\min\text{Coverage}, \text{cover}(\text{NewLHS}) - \max\_\text{spec} \times \max\_\text{reduce})
\]

where

\[
\max\_\text{spec} = \min(\max\text{LHS}\text{size} - |\text{NewLHS}|, |\text{NewAvailableLHS}|)
\]

and

\[
\max\_\text{reduce} = \max_{S \in \text{NewAvailableLHS}} (\text{cover}(\text{CurrentLHS}) - \text{cover}(\text{CurrentLHS} \cup \{S\}))
\]

If \( \text{opt\_confidence} < \text{minConfidence} \), \( Q \) can be pruned from \( \text{NewAvailableRHS} \). Let

\[
\text{opt\_leverage} = \text{support}(\text{NewLHS} \rightarrow Q) - \text{min\_cover} \times \text{cover}(\{Q\})
\]

If \( \text{opt\_leverage} < \text{minLeverage} \), \( Q \) can be pruned from \( \text{NewAvailableRHS} \).

**Proof.** Assume when \( \text{opt\_confidence} < \text{minConfidence} \), \( Q \) cannot be pruned from \( \text{NewAvailableRHS} \). So there exists a rule \( \text{NewLHS} \cup L \rightarrow Q \in \text{solution}(\langle \mathcal{A}, \mathcal{C}, \mathcal{D}, \mathcal{M} \rangle) \) where \( L \subseteq \text{NewAvailableLHS} \) and \( L \neq \emptyset \). Let \( L = \{L_1\} \cup \cdots \cup \{L_l\} \) where \( L_1, \ldots, L_l \in \text{NewAvailableLHS} \), \( 1 \leq l \leq |\text{NewAvailableLHS}| \). Since only \( \max\text{LHS}\text{size} \) of conditions are allowed on antecedent of rules in the solution, \( l \leq \max\text{LHS}\text{size} - |\text{NewLHS}| \) holds. Thus we obtain:

\[
l \leq \min(\max\text{LHS}\text{size} - |\text{NewLHS}|, |\text{NewAvailableLHS}|) = \max\_\text{spec}
\]

\( \text{cover}(\text{NewLHS} \cup L) \) is the number of records covered by \( \text{NewLHS} \) minus the number of records not covered by \( \text{NewLHS} \cup L \), divided by \( |\mathcal{D}| \). Thus the following holds.

\[
\text{cover}(\text{NewLHS} \cup L) = \text{cover}(\text{NewLHS}) - \frac{|\{d \in \mathcal{D} \mid \text{NewLHS} \subseteq \text{conditions}(d) \land \text{NewLHS} \cup L \nsubseteq \text{conditions}(d)\}|}{|\mathcal{D}|}
\]

(19)

For any \( d_1 \in \{d \in \mathcal{D} \mid \text{NewLHS} \subseteq \text{conditions}(d) \land \text{NewLHS} \cup L \nsubseteq \text{conditions}(d)\} \), since \( \text{NewLHS} = \text{CurrentLHS} \cup \{P\} \) and \( \text{NewLHS} \subseteq \text{conditions}(d_1) \), \( \text{CurrentLHS} \subseteq \text{conditions}(d_1) \).
holds. From \( \text{NewLHS} \cup L \not\subseteq \text{conditions}(d_1) \), we obtain \( L \not\subseteq \text{conditions}(d_1) \). Thus \( \text{CurrentLHS} \cup L \not\subseteq \text{conditions}(d_1) \) holds. Hence,

\[
d_1 \in \{ d \mid d \in \mathcal{D} \land \text{CurrentLHS} \subseteq \text{conditions}(d) \land \text{CurrentLHS} \cup L \not\subseteq \text{conditions}(d) \}\]

Thus the following holds.

\[
\{ d \mid d \in \mathcal{D} \land \text{NewLHS} \subseteq \text{conditions}(d) \land \text{NewLHS} \cup L \not\subseteq \text{conditions}(d) \} 
\subseteq
\{ d \mid d \in \mathcal{D} \land \text{CurrentLHS} \subseteq \text{conditions}(d) \land \text{CurrentLHS} \cup L \not\subseteq \text{conditions}(d) \}\]  \tag{20}

According to the definition of \( \text{max}_\text{reduce} \) in (17), for any \( 1 \leq i \leq l \),

\[
\frac{|\{ d \mid d \in \mathcal{D} \land \text{CurrentLHS} \subseteq \text{conditions}(d) \land \text{CurrentLHS} \cup \{L_i\} \not\subseteq \text{conditions}(d) \}|}{|\mathcal{D}|} \leq \text{max}_\text{reduce}
\]

Since \( L = \{L_1\} \cup \cdots \cup \{L_l\} \), and from (18), (20) and (21), we obtain:

\[
\frac{|\{ d \mid d \in \mathcal{D} \land \text{NewLHS} \subseteq \text{conditions}(d) \land \text{NewLHS} \cup L \not\subseteq \text{conditions}(d) \}|}{|\mathcal{D}|} \leq \frac{\sum_{i=1}^{l} |\{ d \mid d \in \mathcal{D} \land \text{CurrentLHS} \subseteq \text{conditions}(d) \land \text{CurrentLHS} \cup \{L_i\} \not\subseteq \text{conditions}(d) \}|}{|\mathcal{D}|} \leq \text{max}_\text{spec} \times \text{max}_\text{reduce}
\]  \tag{22}

From (19) and (22), we obtain:

\[
\text{cover}(\text{NewLHS} \cup L) \geq \text{cover}(\text{NewLHS}) - \text{max}_\text{spec} \times \text{max}_\text{reduce}
\]  \tag{23}

Since \( \text{NewLHS} \cup L \rightarrow Q \in \text{solution}((A,C,D,M)) \), \( \text{cover}(\text{NewLHS} \cup L) \geq \text{minCoverage} \) holds. From (23) and (17), we obtain:

\[
\text{cover}(\text{NewLHS} \cup L) \geq \text{min}\text{cover}
\]

Therefore, according to the subset cover lemma, \( \text{confidence}(\text{NewLHS} \cup L \rightarrow Q) \) satisfies:

\[
\text{confidence}(\text{NewLHS} \cup L \rightarrow Q) = \frac{\text{cover}(\text{NewLHS} \cup L \cup \{Q\})}{\text{cover}(\text{NewLHS} \cup L)} \leq \frac{\text{cover}(\text{NewLHS} \cup \{Q\})}{\text{cover}(\text{NewLHS} \cup L)} = \frac{\text{support}(\text{NewLHS} \rightarrow Q)}{\text{cover}(\text{NewLHS} \cup L)} \leq \frac{\text{support}(\text{NewLHS} \rightarrow Q)}{\text{min}\text{cover}} = \text{opt}\text{confidence} < \text{minConfidence}
\]
This contradicts the proposition that $NewLHS \cup L \to Q$ is in the solution. Therefore $Q$ can be pruned from $NewAvailableRHS$. Accordingly, $	ext{leverage}(NewLHS \cup L \to Q)$ satisfies:

$$
\text{leverage}(NewLHS \cup L \to Q) = \frac{\text{support}(NewLHS \cup L \to Q) - \text{cover}(NewLHS \cup L) \times \text{cover}(\{Q\})}{\text{cover}(NewLHS \cup L \cup \{Q\}) - \text{cover}(NewLHS \cup L) \times \text{cover}(\{Q\})}
$$

This contradicts the proposition that $NewLHS \cup L \to Q$ is in the solution. Therefore $Q$ can be pruned from $NewAvailableRHS$.

Inherent in the selection of pruning rules is a trade-off between the amount of computation required to identify opportunities to prune and the amount of computation saved by applying pruning. The $opt\_confidence$ measure requires little computation to evaluate, but is a very loose upper bound on confidence, and hence is less effective at pruning than a tighter bound would be.

The third pruning rule utilizes a tighter bound on confidence, $opt\_confidence'$ which requires a one-step lookahead to compute. This requires greater computation than $opt\_confidence$, and hence is evaluated only after other pruning rules have failed to prune a node in the search space.

**Pruning 8.** In GRD for $GRDtask = \langle A, C, D, M \rangle$, after the evaluation of the current rule $NewLHS \to Q$, $opt\_confidence'$ is computed by:

$$
op\_confidence' = \frac{\text{support}(NewLHS \to Q)}{\text{min} \_cover'}
$$

where

$$\text{min} \_cover' = \max(\text{minCoverage}, \text{cover}(NewLHS) - \text{max} \_spec \times \text{max} \_reduce')$$

(24)

where

$$\text{max} \_spec = \min(\text{maxLHSsize} - |NewLHS|, |NewAvailableLHS|)$$

and

$$\text{max} \_reduce' = \max_{S \in \text{NewAvailableLHS}} (\text{cover}(NewLHS) - \text{cover}(NewLHS \cup \{S\}))$$

If $opt\_confidence' < \text{minConfidence}$, $Q$ can be pruned from $NewAvailableRHS$. Let

$$\text{leverage'} = \text{support}(NewLHS \to Q) - \text{min} \_cover' \times \text{cover}(\{Q\})$$

If $\text{leverage'} < \text{minLeverage}$, $Q$ can be pruned from $NewAvailableRHS$.

**Proof.** Assume when $opt\_confidence' < \text{minConfidence}$, $Q$ cannot be pruned from $NewAvailableRHS$. So there exists a rule $NewLHS \cup L \to Q \in \text{solution}(\langle A, C, D, M \rangle)$ where $L \subseteq NewAvailableLHS$ and $L \neq \emptyset$. Let $L = \{L_1\} \cup \cdots \cup \{L_l\}$ where $L_1, \cdots, L_l \in NewAvailableLHS, 1 \leq l \leq |NewAvailableLHS|$. Since only maxLHSsize conditions are allowed on antecedent of rules in the solution, $l \leq \text{maxLHSsize} - |NewLHS|$ holds. Thus we obtain:

$$l \leq \min(\text{maxLHSsize} - |NewLHS|, |NewAvailableLHS|) = \text{max} \_spec
$$

(25)

$\text{cover}(NewLHS \cup L)$ is the number of records covered by $NewLHS$ minus the number of records not covered by $NewLHS \cup L$, divided by $|D|$. Thus the following holds.

$$\text{cover}(NewLHS \cup L) = \frac{\text{cover}(NewLHS) - \text{|\{d \in D \land NewLHS \subset \text{conditions}(d) \land NewLHS \cup L \not\subset \text{conditions}(d)\}|}}{|D|}
$$

(26)
According to the definition of $\text{max}_\text{reduce}'$ in (24), for any $1 \leq i \leq l$,

$$\left\{|d|d \in \mathcal{D} \land \text{NewLHS} \subset \text{conditions}(d) \land \text{NewLHS} \cup \{L_i\} \not\subset \text{conditions}(d)\right\}| \leq \text{max}_\text{reduce}'$$

(27)

Since $L = \{L_1\} \cup \cdots \cup \{L_l\}$, and from (25) and (27), we obtain:

$$\frac{\left\{|d|d \in \mathcal{D} \land \text{NewLHS} \subset \text{conditions}(d) \land \text{NewLHS} \cup L \not\subset \text{conditions}(d)\right\}}{|\mathcal{D}|} \leq \text{max}_\text{spec} \times \text{max}_\text{reduce}'$$

(28)

From (26) and (28), we obtain:

$$\text{cover}((\text{NewLHS} \cup L) \not\rightarrow Q) \geq \text{cover}((\text{NewLHS}) - \text{max}_\text{spec} \times \text{max}_\text{reduce}'$$

(29)

Since $\text{NewLHS} \cup L \rightarrow Q \in \text{solution}((\mathcal{A}, \mathcal{C}, \mathcal{D}, \mathcal{M}))$, $\text{cover}((\text{NewLHS} \cup L) \geq \text{minCoverage}$ holds.

From (29) and (24), we obtain:

$$\text{cover}((\text{NewLHS} \cup L) \geq \text{min}_\text{cover}'$$

Therefore, according to the subset cover lemma, $\text{confidence}((\text{NewLHS} \cup L) \rightarrow Q)$ satisfies:

$$\text{confidence}((\text{NewLHS} \cup L) \rightarrow Q) = \frac{\text{cover}((\text{NewLHS} \cup L) \rightarrow Q)}{\text{cover}((\text{NewLHS} \cup L}) \leq \frac{\text{cover}((\text{NewLHS} \cup Q)}{\text{cover}((\text{NewLHS} \cup L}) \leq \frac{\text{support}(\text{NewLHS} \rightarrow Q)}{\text{support}((\text{NewLHS} \rightarrow Q)} \leq \frac{\text{min}_\text{cover}'}{\text{support}((\text{NewLHS} \rightarrow Q)} \leq \text{opt}_\text{confidence'}$$

This contradicts the proposition that $\text{NewLHS} \cup L \rightarrow Q$ is in the solution. Therefore $Q$ can be pruned from $\text{NewAvailableRHS}$. Accordingly, $\text{leverage}((\text{NewLHS} \cup L) \rightarrow Q)$ satisfies:

$$\text{leverage}((\text{NewLHS} \cup L) \rightarrow Q) = \text{support}((\text{NewLHS} \cup L) \rightarrow Q) - \text{cover}((\text{NewLHS} \cup L) \times \text{cover}((\{Q\}))$$

$$\leq \text{cover}((\text{NewLHS} \cup Q}) - \text{cover}((\text{NewLHS} \cup L) \times \text{cover}((\{Q\}))$$

$$\leq \text{cover}((\text{NewLHS} \cup Q}) - \text{min}_\text{cover}' \times \text{cover}((\{Q\}))$$

$$= \text{support}((\text{NewLHS} \rightarrow Q) - \text{min}_\text{cover}' \times \text{cover}((\{Q\}))$$

$$= \text{opt}_\text{leverage'}$$

$$< \text{minLeverage}$$

This contradicts the proposition that $\text{NewLHS} \cup L \rightarrow Q$ is in the solution. Therefore $Q$ can be pruned from $\text{NewAvailableRHS}$. ☐

Although it requires considerable additional computation to evaluate $\text{cover}((\text{NewLHS} \cup \{S\})$ where $S \in \text{NewAvailableLHS}$ for $\text{max}_\text{reduce}'$, pruning by $\text{opt}_\text{confidence'}$ and $\text{opt}_\text{leverage'}$ still improves the efficiency of the search.
6.7. Dynamic Constraint Update

Although the constraints minCoverage, minSupport, and minConfidence are initialized by the user, during search it may be possible to derive tighter constraints on these statistics than those initial values. Such tighter constraints can be used to prune more of the search space or save the evaluation of rules. Based on the theorem of support for rules in the solution, the theorem of coverage for rules in the solution and the theorem of support for rules in the solution related to leverage, whenever a new rule is added to currentSolution at line 19 of the algorithm, all the constraints can be updated at line 21 according to the rules listed below.

1. If minSupport < minCoverage \times minConfidence, then minSupport = minCoverage \times minConfidence.
2. If minCoverage < minSupport, then minCoverage = minSupport.
3. If minSupport < minLeverage, then minSupport = minLeverage.

6.8. Saving Data Access by Identifying Unqualified Antecedents

One of the large computational overheads in rule discovery is accessing the dataset to evaluate the required statistics with respect to each antecedent, consequent, and antecedent-consequent pair. One of the key factors in the success of the Apriori algorithm is its success in minimizing the number of such data accesses that are required. In addition to pruning regions of the search space, another important technique used in GRD to reduce compute time is to save data access. Data access is required to evaluate the cover and support of a rule or a set of conditions. However, data access can be avoided when these values can be derived from other evidence or it can be determined that the values will not satisfy the search constraints. Different saving rules can be adopted at different stages during the search process. Whereas the pruning rules save data access by discarding the region of the search space below a node, the saving rules save data access for a node without removing its branch.

In order to evaluate the number of records covered by set of conditions, the dataset is normally accessed by GRD at least once for each antecedent (NewLHS) and once for the union of the antecedent and consequent. Techniques for saving such data access are directed at avoiding the need to perform one or the other of these computations.

Line 12 in GRD is for saving data access for the rules with NewLHS as antecedent. We give two saving rules. The first is based on the theorem of minimum leverage for the antecedent of rules in the solution.

**Saving 1.** In GRD for GRDtask = (A,C,D,M), if |NewLHS| = maxLHSsize and cover(NewLHS) \times (1 - cover(NewLHS)) < minLeverage, for any Q \in NewAvailableRHS, there is no need to access data to evaluate NewLHS \rightarrow Q, as it is not in the solution.

Please note that although such a NewLHS is not qualified to be the antecedent for a rule in the solution, it cannot be pruned since some of its supersets might have leverage larger than minLeverage. Saving the data access for all rules with NewLHS as the antecedent might prevent application of pruning rules due to the absence of information about cover(NewLHS \cup Q) where Q \in NewAvailableRHS. For this reason, we add the limitation to the above saving rule that |NewLHS| = maxLHSsize, which is the maximum search depth and below which no pruning will be performed. Saving data access at this stage cannot slow down the overall efficiency, as cover(NewLHS \cup Q) is not used for pruning.

The second saving rule is for a NewLHS that covers the whole dataset.

**Saving 2.** In GRD for GRDtask = (A,C,D,M), if |NewLHS| = maxLHSsize, cover(NewLHS) = 1 and minLeverage > 0, for any Q \in NewAvailableRHS, there is no need to access data to evaluate NewLHS \rightarrow Q, as it is not in the solution.
According to the theorem of full cover, the leverage value of any rule with such \( NewLHS \) as the antecedent is 0. When \( \text{minLeverage} > 0 \), such a rule cannot be in the solution. However, such a \( NewLHS \) cannot be pruned since some of its supersets may not cover the whole dataset anymore and thus can make the rule have leverage larger than 0.

6.9. Saving data access by identifying unqualified rules

Line 17 in GRD is for saving data access for the current rule \( NewLHS \rightarrow Q \). We give two saving rules. The first is based on the theorem of minimum confidence for rules in the solution.

**Saving 3.** In GRD for \( GRD\text{task} = (\{A,C,D,M\}) \), for the current rule \( NewLHS \rightarrow Q \), if \( |NewLHS| = \text{maxLHSsize} \) and \( \frac{\text{cover}(Q)}{\text{cover}(NewLHS)} < \text{minConfidence} \), there is no need to access data to evaluate \( NewLHS \rightarrow Q \), as it is not in the solution.

The reason that the saving is adopted instead of pruning under this situation is in the branch below the current \( NewLHS \rightarrow Q \), some of the supersets of \( NewLHS \) with lower values of coverage might make the rule have confidence larger than \( \text{minConfidence} \). While saving data access, it is no longer possible to perform pruning based on the results of the data access. In consequence, the overall efficiency might be slowed down accordingly. Due to this, the constraint \( |NewLHS| = \text{maxLHSsize} \) is added to the above saving rule to ensure that it is applied only at the maximum search depth where no pruning is necessary.

The next saving rule is based on the theorem of minimum leverage for rules in the solution.

**Saving 4.** In GRD for \( GRD\text{task} = (\{A,C,D,M\}) \), for the current rule \( NewLHS \rightarrow Q \), if \( |NewLHS| = \text{maxLHSsize} \) and \( \text{cover}(NewLHS) > 1 - \frac{\text{minLeverage}}{\text{cover}(Q)} \), there is no need to access data to evaluate \( NewLHS \rightarrow Q \), as it is not in the solution.

Any rule as described above has a value for leverage less than \( \text{minLeverage} \). However no pruning should be performed here as some of the supersets of \( NewLHS \) might have lower cover than \( (1 - \frac{\text{minLeverage}}{\text{cover}(Q)}) \) and thus have leverage larger than \( \text{minLeverage} \).

6.10. Saving data access by identifying generalizations with identical statistics

During the evaluation of the current rule \( NewLHS \rightarrow Q \), where \( NewLHS = \text{CurrentLHS} \cup \{P\}, P \in \text{AvailableLHS} \), line 18 in GRD adopts a data access saving rule utilizing the relationship between \( \text{CurrentLHS} \) and \( P \). It is based on the theorem of relation from coverage.

**Saving 5.** In GRD for \( GRD\text{task} = (\{A,C,D,M\}) \), for the current rule \( NewLHS \rightarrow Q \) where \( NewLHS = \text{CurrentLHS} \cup \{P\}, P \in \text{AvailableLHS} \), if \( |NewLHS| = \text{maxLHSsize} \), the number of rules in \( \text{currentSolution} \) is less than \( |\text{coverset}(NewLHS)| \), and \( \text{cover}(\text{CurrentLHS}) = \text{cover}(NewLHS) \), instead of accessing data to evaluate \( NewLHS \rightarrow Q \), check if \( \text{CurrentLHS} \rightarrow Q \) exists in \( \text{currentSolution} \), and if yes, copy all the statistic values of \( \text{CurrentLHS} \rightarrow Q \) to \( NewLHS \rightarrow Q \), otherwise, \( NewLHS \rightarrow Q \) is not in the solution.

Since \( \text{CurrentLHS} \rightarrow Q \) is investigated before \( NewLHS \rightarrow Q \) in GRD, and they share the same statistic values, \( NewLHS \rightarrow Q \) will be in \( \text{currentSolution} \) if and only if \( \text{CurrentLHS} \rightarrow Q \) is in \( \text{currentSolution} \). Due to the same reasons as in the above subsection, we add \( |NewLHS| = \text{maxLHSsize} \) in the saving rule to make sure that application of the saving rule cannot slow down the overall efficiency. If the number of rules in \( \text{currentSolution} \) is greater than \( |\text{coverset}(NewLHS)| \), searching these rules might be less efficient than accessing \( \text{coverset}(NewLHS) \) to compute \( \text{cover}(NewLHS \cup Q) \).
7. Proof-of-concept experiments

We investigate the computational efficiency of GRD search for rules that optimize leverage. Experiments are performed on ten large datasets, nine from the UCI ML and KDD repositories (Blake and Merz, 2001; Bay, 2001) and one market-basket dataset used in previous association rule discovery research (Kohavi et al., 2000; Zheng et al., 2001). These datasets are listed in Table II. In our experiments for all the datasets, all the conditions available are allowed both in the antecedent and consequent of rules. Numeric attributes were discretized into three sub-ranges, each containing as close as possible to one third of the records.

It might be thought that traditional association rule techniques could be applied to this problem by first finding all frequent itemsets and then generating all rules from those itemsets, sorting them on leverage, and discarding all but the top $n$. We evaluate the feasibility of this approach by applying Borgelt’s (2000) efficient implementation of Apriori. We do not include recent alternatives to Apriori in this study as it has been argued elsewhere (Zheng et al., 2001) that they suffer the same performance degradation as Apriori on tasks for which minimum support is set to a level that results in excessively large numbers of frequent itemsets, as is the case for the current task with respect to many real world datasets.

A difficulty with applying the frequent itemset approach to discovering the $n$ rules with highest leverage is that there does not appear to be any way to determine apriori what minimum support level to employ. However, it follows from the definitions of leverage and support that \( \text{leverage}(X \rightarrow Y) \leq \text{support}(X \rightarrow Y) \). Hence, if rules $R$ are derived from the set of itemsets $\{i \mid \text{support}(i) \geq nth\}$, where $nth$ is the $nth$ highest leverage a rule in $R$, then it follows that the $n$ rules with highest leverage derived from those itemsets are the $n$ rules with highest leverage that would be derived without a minimum support constraint. Using this insight, an iterative process could repeatedly generate frequent itemsets at gradually lowering values of $\text{minSupport}$ and generate rules $R$ therefrom until $\text{leverage}(R_n) \geq \text{minSupport}$, where $\text{leverage}(R_n)$ is the $nth$ highest value of leverage of a rule in $R$. To give an indication of a lower bound on the overheads of such an approach, we use Apriori with $\text{minSupport}$ set to the minimum value for leverage of the 1000 rules with highest leverage as discovered by GRD.

It might be thought that the earlier OPUS (Webb, 1995), Max-Miner (Bayardo, 1998) and Dense-Miner (Bayardo et al., 2000) algorithms, upon which GRD builds, should also be comparators against which GRD is evaluated. However, this is not feasible, as those algorithms require that the consequent be restricted to a single pre-specified value, and hence are not capable of performing the generalized rule discovery task. It is, indeed, this very limitation that GRD has been designed to overcome.

In all the experiments, GRD seeks the top 1000 rules on leverage within the constraints of the maximum number of conditions in the antecedent of a rule is 4 and the maximum number of conditions in the consequent of a rule is 1. The specific constraint on the size of the antecedent was chosen because it is the default for the implementation of Apriori that we also used in this study. The same
Table III. Efficiency of GRD and Apriori on real world datasets

<table>
<thead>
<tr>
<th>Datasets</th>
<th>CPU time</th>
<th>GRD</th>
<th>Apriori</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>hr:min:sec</td>
<td>rules evaluated</td>
<td>CPU time</td>
</tr>
<tr>
<td>BMS-WebView-1</td>
<td>0:0:4</td>
<td>91,244</td>
<td>0:0:1</td>
</tr>
<tr>
<td>connect-4</td>
<td>0:0:55</td>
<td>94,682</td>
<td>1:0:2</td>
</tr>
<tr>
<td>covtype</td>
<td>0:8:3</td>
<td>155,489</td>
<td>30:2:55</td>
</tr>
<tr>
<td>ipums.la.99</td>
<td>0:0:29</td>
<td>104,582</td>
<td>Not enough memory</td>
</tr>
<tr>
<td>kddcup98</td>
<td>1:31:30</td>
<td>6,259,666</td>
<td></td>
</tr>
<tr>
<td>mush</td>
<td>0:0:1</td>
<td>12,892</td>
<td>0:0:4</td>
</tr>
<tr>
<td>pendigits</td>
<td>0:0:1</td>
<td>50,733</td>
<td>0:0:3</td>
</tr>
<tr>
<td>shuttle</td>
<td>0:0:1</td>
<td>10,029</td>
<td>0:0:4</td>
</tr>
<tr>
<td>splice junction</td>
<td>0:0:6</td>
<td>2,160,784</td>
<td>0:0:38</td>
</tr>
<tr>
<td>ticdata2000</td>
<td>0:0:33</td>
<td>187,247</td>
<td>0:38:47</td>
</tr>
</tbody>
</table>

maximum antecedent and consequent size used for GRD were also used for Apriori. The experiments were performed on a Linux server with 2 CPUs each 933MHz in speed, 1.5G RAM, and 4G virtual memory.

Table III shows the efficiency of GRD and Apriori on the ten large datasets. For GRD this table presents the compute time in hours, minutes, and seconds; the number of rules evaluated; and the minimum leverage for a rule in the top 1000 rules on leverage. For Apriori this table lists the compute time for frequent itemset generation in hours, minutes and seconds, and the number of itemsets generated.

Comparing CPU times we can see that on every dataset other than BMS-WebView-1, for which compute times are extremely small, GRD requires less compute time than Apriori. For kddcup98 Apriori runs out of memory when generating itemsets. This supports previous analyses of the inefficiency of the frequent itemset strategy for dense datasets (Bayardo, 1998). It also suggests that GRD provides a more widely applicable approach to the generalized rule discovery problem than Apriori when the user seeks to discover a limited number of rules that optimize an interestingness measure.

8. Computational complexity and scalability

The worst-case complexity of GRD is exponential on the number of conditions that are available for inclusion in the antecedent and consequent, as in the worst case no pruning is possible and the entire rule space must be explored. If there are $k$ conditions that may appear in an antecedent then the space of possible antecedents is $2^k$. If any one of these $k$ conditions may also appear in the consequent, then the space of possible rules is of order $O(2^{k+1})$, despite a restriction that a condition may not appear in both the antecedent and conclusion of the same rule. The efficiency of the algorithm depends critically, then, upon the effectiveness of the pruning mechanisms for a given task. As the experiments above have demonstrated, for a number of larger real-world datasets, efficient search by leverage is a reality.

This raises the question, however, of how well will the algorithm scale to even larger datasets. A fundamental limitation of the current algorithm is that it requires that all data be retained in memory during search, as it requires frequent assessment of relevant statistics. With current hardware, this restricts application of the algorithm to datasets of size measured in gigabytes. It is conceivable, however, that a variant of the algorithm could be created that uses the strategy developed for Dense-Miner (Bayardo et al., 2000), performing breadth-first search with two stages at each level, the first generating candidate rules and the second evaluating them all in a single pass through the data.

In theory, the computation required by GRD should increase linearly with respect to the size of the data, as the size of the data affect directly only the process of counting the statistics used
to evaluate a rule. To assess this, we took the largest of our datasets, covtype, and formed sample datasets containing the first 25%, 50%, 75% and 100% of the data. We also formed larger datasets by appending each of these sample datasets to the original, resulting in four further datasets containing 125%, 150%, 175% and 200% of the original amount of data, albeit with some records duplicated. Figure 4 plots the size of these datasets against the compute time required to perform the task described in Section 7.

Inspection of the resulting graph reveals that the points 25%, 50%, 75% and 100% appear to form a curve that it slightly super-linear. However, it is clear that this effect is not due solely to the direct increased costs of processing the additional data, as the processing time for 200%, in which all data points are duplicated, is almost exactly twice that of 100%. Rather, due to the characteristics of this data, as the number of data points increases up to 100%, there are more combinations of conditions that achieve sufficient support to make them credible candidates during search. Hence the number of rules explored increases. Figure 5 plots the number of rules explored at each data set size. As can be seen, there is an increase in the number of rules up until the stage when 100% of the data is used. When data elements are simply duplicated, however, no more rules need to be considered. In this case, the increase in execution time reflects only an increase in the time taken to evaluate the statistics for each rule that is considered.

To assess whether the number of rules evaluated can always be expected to increase as the number of unique data points increases, we repeated the experiment with the next largest data set, ipums.la.99. As the effect of duplicating data elements does not appear to need further investigation, we used only data sets containing 25%, 50%, 75% and 100% of the data. The compute time and number of rules evaluated are plotted in Figures 6 and 7. It is evident that there is no simple relationship between data set size and the number of rules that need to be explored. As a result, the amount of time taken varies at different rates over different increases in data size. Processing all the data takes 29 seconds which is less than 4 times the 8 seconds taken to process 25% of the data. We can expect the increase on average to be linear on the number of records, but relationships within the data may mean that additional records result in the need to explore either more or fewer potential rules, resulting in concomitant increases or decreases in computation.
Rule discovery techniques have evolved within two distinct paradigms, classification rule discovery and association rule discovery. The focus of classification rule discovery on finding small sets of highly predictive rules that jointly cover all available data is not well suited to contexts in which the primary objective is to gain insight into interactions within the data. While association rule discovery is directly aimed at this objective, its support-confidence framework also limits its applicability. In particular, where minimum support is not a relevant criterion for selecting rules, the frequent itemset strategies that have been developed for association rule discovery risk failing to identify interesting rules.

We have presented a new framework for rule discovery which we call generalized rule discovery. Generalized rule discovery subsumes association rule discovery and the form of rule search generally employed to discover individual rules within classification rule discovery. We have presented a scalable algorithm for generalized rule discovery. The computational complexity of the algorithm is linear with
Figure 7. Number of rules evaluated vs data set size for variants of the ipums.la.99 dataset

respect to data set size, although different sets of data imply different search spaces which may require
greater or lesser computation to explore. We have demonstrated that GRD can efficiently solve an
interesting class of generalized rule discovery task, finding the 1000 rules with the highest leverage. We
have further demonstrated that it is difficult to tackle this task using the frequent itemset framework
pioneered within the association rule discovery paradigm.

An important benefit of the generalized rule discovery framework is the ability to impose and
gain computational benefit from contraints on the maximum number of rules to discover. A common
complaint levelled against association rule discovery is that the user cannot control how many rules are
returned. The number of rules returned is controlled indirectly by the minimum support constraint.
Small changes in this constraint can transform a problem for which very few rules are discovered
to one for which millions of rules are discovered. In the generalized rules discovery framework the
user can place an upper limit $maxRules$ on the number of rules to discover. This constraint ensures
that the user is only presented with the most interesting rules according to the specified measure of
interestinless. It also speeds up the rule discovery task by allowing pruning of areas of the search
space that cannot contain one of the $maxRules$ most interesting rules.

This work demonstrates the efficiency of the GRD algorithm for one type of interesting rule dis-
covery task. It falls to future research to identify further sets of interesting constraints for generalized
rule discovery and develop pruning rules that will facilitate their efficient solution.

Appendix

A. Proof of sufficiency and necessity of update strategy

The following theorems prove that the update strategy for $currentSolution$ is sufficient and necessary
to ensure that on termination $currentSolution = solution((A, C, D, M))$.

Theorem 12 (Sufficiency of $currentSolution$). Suppose $GRDtask = (A, C, D, M)$. For any $S \subseteq
\{X \rightarrow Y \mid X \subseteq A \land Y \subseteq C\}$, let $currentSolution = solution((A, C, D, M \land X \rightarrow Y \in S))$. For any
$s \in solution((A, C, D, M)) \cap S$, $s \in currentSolution$ holds.

This follows directly from (1).

Theorem 13 (Necessity of $currentSolution$). Suppose $GRDtask = (A, C, D, M)$. For any $S \subseteq
\{X \rightarrow Y \mid X \subseteq A \land Y \subseteq C\}$, let $currentSolution = solution((A, C, D, M \land X \rightarrow Y \in S))$. For any
$s \in currentSolution$, $s \in solution((A, C, D, M)) \cap S$ holds.
Theorem 15 (Necessity of addition to currentSolution). Assume there exists a rule \( s_1 \) satisfies \( s_1 \in \text{currentSolution} \) but \( s_1 \notin \text{solution}(\langle A, C, D, M \rangle) \cap S \). So \( s_1 \notin \text{solution}(\langle A, C, D, M \rangle) \) or \( s_1 \notin S \) holds. If \( s_1 \notin \text{solution}(\langle A, C, D, M \rangle) \), then \( s_1 \notin \text{solution}(\langle A, C, D, M \land X \rightarrow Y \in S \rangle) \), contradicting \( s_1 \in \text{currentSolution} \); if \( s_1 \notin S \), obviously \( s_1 \notin \text{solution}(\langle A, C, D, M \land X \rightarrow Y \in S \rangle) \), contradicting \( s_1 \in \text{currentSolution} \). Therefore \( s_1 \) does not exist. \( \Box \)

Due to the following theorem, it is not necessary to keep track of \( S \). Rather, it is sufficient to update \( \text{currentSolution} \) as each rule is added to \( S \).

Theorem 14 (Sufficiency of addition to currentSolution). Suppose GRDtask = \( \langle A, C, D, M \rangle \). For any \( S \subseteq \{ X \rightarrow Y \mid X \subseteq A \land Y \subseteq C \} \) and rule \( r \notin S \), let \( \text{currentSolution} = \text{solution}(\langle A, C, D, M \land X \rightarrow Y \in S \rangle) \) and \( \text{currentSolution}' = \text{solution}(\langle A, C, D, M \land X \rightarrow Y \in S \cup \{ r \} \rangle) \). The following holds.
\[
\text{solution}(\langle A, C, D, M \rangle) \cap (S \cup \{ r \}) \subseteq \text{solution}(\langle A, C, D, M \land X \rightarrow Y \in S \cup \{ r \} \rangle)
\]
Proof. Assume there exists a rule \( s_1 \) that satisfies
\[
s_1 \in \text{solution}(\langle A, C, D, M \land X \rightarrow Y \in \text{currentSolution} \cup \{ r \} \rangle)
\]
but \( s_1 \notin \text{currentSolution}' \) which is
\[
s_1 \notin \text{solution}(\langle A, C, D, M \land X \rightarrow Y \in S \cup \{ r \} \rangle)
\]
According to the theorem of sufficiency of currentSolution, the following holds.
\[
\text{solution}(\langle A, C, D, M \rangle) \cap (S \cup \{ r \}) \subseteq \text{solution}(\langle A, C, D, M \land X \rightarrow Y \in S \cup \{ r \} \rangle)
\]
From (32) and (33), we obtain
\[
s_1 \notin \text{solution}(\langle A, C, D, M \rangle) \cap (S \cup \{ r \})
\]
From (31) we obtain \( s_1 \in \text{currentSolution} \cup \{ r \} \). Let us first assume \( s_1 \in \text{currentSolution} \), which is \( s_1 \in \text{solution}(\langle A, C, D, M \land X \rightarrow Y \in S \rangle) \). According to the theorem of necessity of currentSolution, \( s_1 \in \text{solution}(\langle A, C, D, M \rangle) \cap S \) holds. Thus \( s_1 \in S \). So \( s_1 \in S \cup \{ r \} \). So \( s_1 \in \text{solution}(\langle A, C, D, M \rangle) \cap (S \cup \{ r \}) \) holds, contradicting (34).

Now let us assume \( s_1 \in \{ r \} \). This means \( s_1 \in S \cup \{ r \} \). From (31) and according to the theorem of necessity of currentSolution, \( s_1 \in \text{solution}(\langle A, C, D, M \rangle) \cap (\text{currentSolution} \cup \{ r \}) \) holds. Thus \( s_1 \in \text{solution}(\langle A, C, D, M \rangle) \). From \( s_1 \in S \cup \{ r \} \), we obtain \( s_1 \in \text{solution}(\langle A, C, D, M \rangle) \cap (S \cup \{ r \}) \) holds, contradicting (34). Thus \( s_1 \) does not exist. It follows that (30) holds. \( \Box \)

Actually, the necessity of addition to currentSolution also holds, as shown in the following theorem.

Theorem 15 (Necessity of addition to currentSolution). Suppose GRDtask = \( \langle A, C, D, M \rangle \). For any \( S \subseteq \{ X \rightarrow Y \mid X \subseteq A \land Y \subseteq C \} \) and rule \( r \notin S \), let \( \text{currentSolution} = \text{solution}(\langle A, C, D, M \land X \rightarrow Y \in S \rangle) \) and \( \text{currentSolution}' = \text{solution}(\langle A, C, D, M \land X \rightarrow Y \in S \cup \{ r \} \rangle) \). The following holds.
\[
\text{solution}(\langle A, C, D, M \rangle) \cap (S \cup \{ r \}) \subseteq \text{solution}(\langle A, C, D, M \land X \rightarrow Y \in \text{currentSolution} \cup \{ r \} \rangle)
\]
Proof. For any \( s \in \text{currentSolution}' \), which means \( s \in \text{solution}(\langle A, C, D, M \land X \rightarrow Y \in S \cup \{ r \} \rangle) \), according to the theorem of necessity of currentSolution, \( s \in \text{solution}(\langle A, C, D, M \rangle) \) holds. Thus
\[
s \in \text{solution}(\langle A, C, D, M \rangle)
\]
and \( s \in S \cup \{ r \} \) holds. Let us first assume \( s \in S \). From (36), \( s \in \text{solution}(\langle A, C, D, M \rangle) \cap S \). According to the theorem of sufficiency of currentSolution, \( s \in \text{solution}(\langle A, C, D, M \land X \rightarrow Y \in S \rangle) = \text{currentSolution} \). Thus \( s \in \text{currentSolution} \cup \{ r \} \). Now let us assume \( s \in \{ r \} \). Obviously \( s \in \text{currentSolution} \cup \{ r \} \) holds. Therefore we obtain
\[
s \in \text{currentSolution} \cup \{ r \}
\]
From (36) and (37), \( s \in \text{solution}(\langle A, C, D, M \rangle) \cap (\text{currentSolution} \cup \{ r \}) \). According to the theorem of sufficiency of currentSolution, we obtain \( s \in \text{solution}(\langle A, C, D, M \land X \rightarrow Y \in \text{currentSolution} \cup \{ r \} \rangle) \). Thus (35) holds. \( \Box \)
Table IV. The generalized rule discovery algorithm without pruning

Algorithm: GRD'((CurrentLHS, AvailableLHS, AvailableRHS)
1: SoFar := ∅
2: for all P in AvailableLHS do
3: NewLHS := CurrentLHS ∪ {P}
4: NewAvailableLHS := SoFar
5: if P in AvailableRHS then
6: NewAvailableRHS := AvailableRHS - P
7: else
8: NewAvailableRHS := AvailableRHS
9: end if
10: for all Q in NewAvailableRHS do
11: if insolution(NewLHS ∪ Q, (A,C,D,M ∧ X → Y ∈ currentSolution ∪ {NewLHS → Q})) then
12: add NewLHS ∪ Q to currentSolution
13: remove from currentSolution any rule
14: end if
15: end for
16: if NewAvailableLHS ≠ ∅ and NewAvailableRHS ≠ ∅ then
17: GRD'(NewLHS, NewAvailableLHS, NewAvailableRHS)
18: end if
19: SoFar := SoFar ∪ {P}
20: end for

B. Correctness of the GRD algorithm without pruning

By removing all the pruning in GRD, we get the GRD' algorithm shown in Table IV. The correctness of GRD' is proven by its uniqueness and completeness. We first prove the completeness of GRD' in the following theorem.

**Theorem 16 (Completeness).** Suppose GRDtask = (A,C,D,M). For any CurrentLHS, AvailableLHS ⊆ A and AvailableRHS ⊆ C satisfying AvailableLHS ≠ ∅, AvailableRHS ≠ ∅, CurrentLHS ∩ AvailableLHS = ∅ and CurrentLHS ∩ AvailableRHS = ∅, let AvailableLHS = {P₁, ⋯ , Pₙ} where n ≥ 1 and P₁, ⋯ , Pₙ ∈ A, let SEARCHED(CurrentLHS, {P₁, ⋯ , Pₙ}, AvailableRHS) denote the set of rules GRD'(CurrentLHS, AvailableLHS, AvailableRHS) has searched after the loop from P₁ to Pₙ at line 2 where 1 ≤ m ≤ n, the following holds.

SEARCHED(CurrentLHS, {P₁, ⋯ , Pₙ}, AvailableRHS) =
{CurrentLHS ∪ L → {Q}|L ≠ ∅ ∧ L ⊆ {P₁, ⋯ , Pₙ} ∧ Q ∈ AvailableRHS ∧ Q ≠ L}

Proof. The proof is by induction on m. For the basis, it is readily seen that the theorem is true when m = 1, that is,

SEARCHED(CurrentLHS, {P₁}, AvailableRHS) =
{CurrentLHS ∪ {P₁} → {Q}|Q ∈ AvailableRHS ∧ Q ≠ P₁}

For the inductive hypothesis, assume that the theorem is true for 1 ≤ m ≤ k. We will prove that the theorem is true for m = k + 1. When GRD' loops at line 2 until Pₖ₊₁, SEARCHED(CurrentLHS, {P₁, ⋯ , Pₖ, Pₖ₊₁}, AvailableRHS) is composed of two parts, one is SEARCHED(CurrentLHS, {P₁, ⋯ , Pₖ}, AvailableRHS), and the other is the set of rules GRD' searches from lines 3 to 19 for Pₖ₊₁. For Pₖ₊₁, from lines 3 to 15, GRD' searches

{CurrentLHS ∪ {Pₖ₊₁} → {Q}|Q ∈ AvailableRHS ∧ Q ≠ Pₖ₊₁}
and at line 17, $GRD'(CurrentLHS \cup \{P_{k+1}\}, \{P_1, \ldots, P_k\}, AvailableRHS)$ is called since at this point $SoFar = \{P_1, \ldots, P_k\}$. Applying the inductive hypothesis, we obtain:

$$SEACHED(CurrentLHS, \{P_1, \ldots, P_k, P_{k+1}\}, AvailableRHS)$$

$$= SEACHED(CURRENTLHS, \{P_1, \ldots, P_k\}, AvailableRHS)$$

$$\cup \{CurrentLHS \cup \{P_{k+1}\} \rightarrow \{Q\}| Q \in AvailableRHS \land Q \neq P_{k+1}\}$$

$$\cup SEACHED(CURRENTLHS \cup \{P_{k+1}\}, \{P_1, \ldots, P_k\}, AvailableRHS)$$

$$= SEACHED(CURRENTLHS, \{P_1, \ldots, P_k\}, AvailableRHS)$$

$$\cup \{CurrentLHS \cup \{P_{k+1}\} \rightarrow \{Q\}| Q \in AvailableRHS \land Q \neq P_{k+1}\}$$

$$\cup \{CurrentLHS \cup \{P_{k+1}\} \cup L_1 \rightarrow \{Q\}| L_1 \neq \emptyset \land L_1 \subseteq \{P_1, \ldots, P_k\}$$

$$\land Q \in AvailableRHS \land Q \notin L_1 \cup \{P_{k+1}\}\}$$

$$\cup \{CurrentLHS \cup \{P_{k+1}\} \cup L_1 \rightarrow \{Q\}| L_1 \subseteq \{P_1, \ldots, P_k\}$$

$$\land \land Q \notin L_1 \cup \{P_{k+1}\}\}$$

$$= \{CurrentLHS \cup L \rightarrow \{Q\}| L \neq \emptyset \land L \subseteq \{P_1, \ldots, P_k\} \land Q \in AvailableRHS \land Q \notin L\}$$

$$\cup \{CurrentLHS \cup \{P_{k+1}\} \cup L_1 \rightarrow \{Q\}| L_1 \subseteq \{P_1, \ldots, P_k\} \land Q \in AvailableRHS$$

$$\land \land Q \notin L_1 \cup \{P_{k+1}\}\}$$

$$= \{CurrentLHS \cup L \rightarrow \{Q\}| L \neq \emptyset \land L \subseteq \{P_1, \ldots, P_k, P_{k+1}\} \land Q \in AvailableRHS \land Q \notin L\}$$

proving the theorem.

For any $GRD_{task} = \langle A, C, D, M \rangle$, according to the theorem of sufficiency of addition to current Solution, the variable current Solution at line 12 in $GRD'$ always records solution(\(\langle A, C, D, M \land X \rightarrow Y \in S \cup \{r\}\rangle\)) where $S$ is the search space considered so far and $r$ is the current rule being considered. According to its completeness, when it finishes, $GRD'$ has searched all the rules, thus what the variable current Solution records now is solution(\(\langle A, C, D, M \rangle\)). This proves the uniqueness of $GRD'$.

References


Han, J., J. Pei, and Y. Yin: 2000, ‘Mining Frequent Patterns without Candidate Generation’. In: Proc. 2000 ACM-SIGMOD Int. Conf. on Management of Data (SIGMOD’00). Dallas, TX, pp. 1–12.


Pei, J., J. Han, and R. Mao: 2000, ‘CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets’. In: Proc. 2000 ACM-SIGMOD Int. Workshop on Data Mining and Knowledge Discovery (DMKD’00). Dallas, TX, pp. 21–30.


