# The Hows, Whys, and Whens of Constraints in Itemset and Rule Discovery

Roberto J. Bayardo
IBM Almaden Research Center
bayardo@alum.mit.edu
http://www.almaden.ibm.com/cs/people/bayardo/

**Abstract.** Many researchers in our community (this author included) regularly emphasize the role constraints play in improving performance of data-mining algorithms. This emphasis has led to remarkable progress -- current algorithms allow an incredibly rich and varied set of hidden patterns to be efficiently elicited from massive datasets, even under the burden of NP-hard problem definitions and disk-resident or distributed data. But this progress has come at a cost. In our single-minded drive towards maximum performance, we have often neglected and in fact hindered the important role of discovery in the knowledge discovery and data-mining (KDD) process. In this paper, I propose various strategies for applying constraints within algorithms for itemset and rule mining in order to escape this pitfall.[1]

## 1 Introduction

Constraint-based pattern mining is the process of identifying all patterns in a given dataset that satisfy the specified constraints. There are many types of patterns we may wish to explore, depending on the data or its expected use. To name only a few, we have itemsets, sequences, episodes, substrings, rules, trees, cliques, and so on. The important aspect of constraint-based mining is not so much the specific patterns being identified, but the fact that we would like to identify *all* of them subject to the given constraints. This task of *constraint-based mining* is in contrast to *heuristic* pattern mining which attempts only to identify patterns which are likely (but not guaranteed) to be good according to certain criteria. A third task which I will touch upon only briefly, *optimization-based* pattern mining, attempts to identify only those patterns that are guaranteed to be (among the k-) best according to certain metrics.

While many may assign constraint-based mining a high face value solely from plethora of research on the topic, it is illustrative to take a step back and contemplate ***why*** it is a task worthy of our interest. Indeed, long before the "association rule" was a household name, heuristic pattern miners were proving extremely useful in machine learning circles. In fact, heuristic rule miners, which include decision tree ("divide and conquer") and covering ("separate and conquer") algorithms, remain essential components in the analyst's toolbox. I witnessed a growing interest in constraint-based min-

1 My use of he informal "I" rather than the typical "we" is to emphasize this paper is a personal position statement, along with a view of existing research in light of my position.

ing once heuristic machine learning approaches gained reasonably widespread use in practice. The white-box nature of decision tree and other rule-based models were being used directly for end-user understanding of the data, even though they were not specifically intended for that purpose.[1] Use of these rule-based models for understanding led to questions such as the following:

- Do these rules capture and convey the "essence" of the relationship(s) in my data?
- Are there better rules (and who gets to define better)?

Note that each of these questions is open to some amount of subjective interpretation. But this is the point: the analyst is typically involved in *knowledge discovery* in which subjective and difficult to formalize notions of "goodness" are guiding the process, not simply data mining in which an algorithm follows a deterministic procedure to extract patterns that may or (more often) may not be of interest. Provided that constraints are used sensibly (and what "sensibly" means is the subject of this paper), constraint-based mining fosters discovery by providing the analyst with a broad result set capable of concretely answering a far wider set of questions than one that is heuristically determined.

A theme of this paper is that there are different phases of the knowledge discovery process in which we can exploit constraints, and the specific use of constraints should be dependent on ***when*** (in what phase) we are using them. During the mining phase, I argue that constraints should be *discovery preserving.* That is, they should filter out only those results that are *highly unlikely* to ever be of interest to the analyst. This admittedly informal notion of preserving discovery is in stark contrast to other proposals that envision query languages for constraint-based mining in which every imaginable constraint is enforced directly by the mining phase. The problem with this alternate view is simply that the analyst rarely knows the specific results of interest a priori (no pun intended). Constraints should therefore be used during the mining phase primarily for performance tractability. Discovering the precise results of interest is best left for post-processing of the mining results through interactive interfaces involving visualization, browsing, and ranking.

Recall that optimization-based pattern discovery forms an interesting middle-ground between the heuristic and constraint-based approaches: unlike heuristic approaches, it provides guarantees on result quality. Unlike constraint-based approaches, it provides these guarantees without requiring the extraction of all patterns matching the constraints, the number of which can be enormous. While these are desirable attributes, once again we are confronted with the question of what makes one rule better than the other. Optimization-based approaches allow no ambiguity on the part of the analyst since the ranking function is part of the input, if not hard-coded into the algorithm itself. Should an optimization-based approach be required (for example it is possible the pattern space is simply too large for constraints alone), I argue it is desir-

1 It is therefore ironic that association rule miners are now commonly used in building general classification models, even though originally this was not their intended use!

able for the approach to provide some ability to select and adjust the ranking criteria post-mining [6]. It is tempting to view an optimization criteria as itself just another constraint to be enforced by a constraint-based miner. Viewed as such, an optimization criteria is actually a constraint on the set of patterns rather than a constraint on the properties of the individual patterns. I believe this distinction is important enough to justify treating optimization-based approaches as separate from constraint-based ones.

As researchers, once we are convinced why something is useful, we become obsessed with ***how*** we can achieve it. And with constraint-based mining, the how part is particularly interesting due to huge computational challenges. Many constraint-based mining tasks can be proven NP-hard through reductions from problems such as constraint satisfaction, hitting set, prime implicant, and so on. Worse, the datasets involved often attain volumes beyond which standard data management strategies can efficiently cope. Then there is the issue of ensuring the results of our algorithms have statistical merit. This combination of search, data management, and statistical issues has provided ample research fodder for our community.

I cannot hope to even begin to address all interesting aspects of the hows in constraint-based mining in this short paper, but I will discuss some (often neglected) issues that I feel fit with in the context of discovery preservation. While much of what remains to be discussed applies to pattern mining in general, for concreteness sake, I focus in particular on itemsets and association rules. An itemset is simply a set of values appearing in a given dataset. An association rule is itself an itemset along with additional information specifying the division of items into antecedent and consequent subsets. The seminal work on association rule mining produced algorithms employing two distinct phases: (1) mine the frequent itemsets from the data, (2) output the rules of interest from them. While this two-phase approach was for the most part an operational detail of the mining algorithm, researchers (again, this author included) have been eager to build on only the first phase as if itemsets themselves are the output desired by the end user. I am quick to agree that itemsets are indeed *sometimes* the artifact of interest in data-mining. But that said, I believe, by and large, that the desired outcome of mining is more often *rules* since they express easy to interpret relationships between dataset elements that itemsets alone do not.

Luckily, many itemset constraints are themselves useful rule constraints, thus work in constraint-based itemset mining often has direct applications in constraint-based rule mining. There are, however, many constraints that are specific to rules such as bounds on confidence, lift, and other measures of predictive accuracy, and they have gone virtually ignored outside of result post-processing. To be fair, another reason rule-specific constraints have been ignored is that they do not fall into any of the convenient constraint classes that have been found to be easily enforceable during mining. But the fact is that many of these rule constraints can be broken down into constituents that do fall into these classes. I will overview previous work in which properties of these constituents have been exploited for effective enforcement during mining *given appropriate structuring of the search*. That said, coming back to my original thesis, we typically would not want to enforce arbitrary rule constraints during mining to avoid hindering discovery. I therefore provide examples of rule constraints that can be

regarded as discovery preserving, along with a framework for their enforcement during mining.

## 2 Constraints in the Discovery Process

It is well-known that knowledge discovery is a multi-phase and iterative process [11]. The data preparation and data-mining stages are often the most costly in terms of compute overhead. Thus, if possible, iteration should be restricted to subsequent phases (such as post-processing) in which it can be performed quickly. In the context of pattern mining, the role of the data-mining algorithm should be to transform the (preprocessed) data-set into a representation that allows for interactive browsing, ranking, and querying. "Interactive" means that the effects of changing a parameter, for example via a graphical control, are almost instantaneous. The following figure depicts this view.

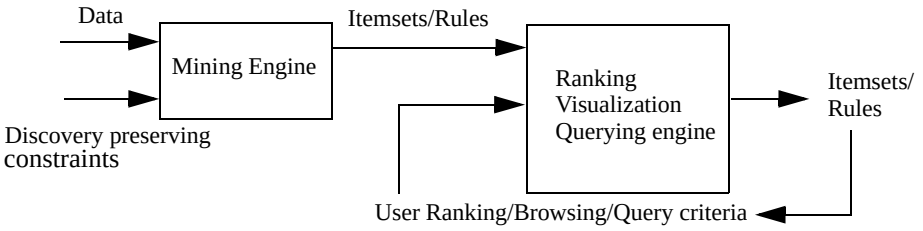


**Fig. 1.** Idealized View of the Mining Process

In some cases the input dataset may be sufficiently compact and the mining sufficiently trivial to allow the data-mining algorithm to be re-applied in real time to support interactivity. Mining caches can be used to further improve response [15,17], though I have doubts that cache hit rates will be significant enough for this to be of much use in practice.

More often, an intermediate representation is required to satisfy interactive response requirements. In the case of constraint-based rule and itemset mining, this intermediate representation is typically some collection of itemsets with their associated support values. For some datasets it might be possible to precompute the support of all possible itemsets and store them in an indexed database. However, most non-trivial datasets have enough items to make this impractical, as the number of itemsets increases exponentially with the number of items. A solution is to apply constraints to reduce the size of the mining result and the time required to obtain it, preferably without excluding patterns that are of interest to the analyst. I argue that a good mining engine constraint has the following properties:

1. It is **tweakable**: post-mining, if the constraint is parameterized, the parameter should be adjustable without requiring expensive processing such as scanning or re-mining the original dataset.

2. It **provides efficiency**: applying the constraint should make the algorithm run significantly more efficiently. At this phase we are more concerned with using constraints for achieving tractability, and not necessarily in speeding up mining by a small constant.
3. It **preserves discovery**: the constraint, if it limits the sets of questions the analyst may efficiently pose during post-processing, should eliminate only those questions that are unlikely to be of value.

Properties 1 and 2 allow for the system itself to specify constraints automatically to ensure tractability of the mining run. The user is then able to efficiently adjust the constraints after the fact if necessary. Property 3 implies that the system has a low probability of excluding patterns that may have otherwise been found interesting by the user.

Property 3 is clearly the most subjective. Indeed, any pattern elimination can probably be rationalized as eliminating something useful for *some* purpose. However there are some constraints that do satisfy these properties in most settings. One example is a very low setting of minimum support. (1) Minimum support can be easily adjusted upwards post-mining without going back to the original dataset. One only needs to filter (or ignore) those itemsets whose supports falls below the modified limit. (2) Minimum support has been proven to provide significant boosts in efficiency during mining, even at relatively low settings. (3) Minimum support, provided it can be set low enough, preserves discovery since results with extremely low support are unlikely to be statistically valid.

Is minimum support enough? I feel it is safe to say that for "market-basket" and other sparse datasets, the answer is wholeheartedly yes. In fact, minimum support as exploited by the earliest of association rule miners (such as Apriori) is often entirely sufficient. In figure 2, I reprint with permission two graphs from a recent workshop on frequent itemset mining implementations (FIMI-03 [12]) in which participants submitted implementations for apples-to-apples comparison on a variety of datasets. For the sparse datasets, Borgelt's Apriori implementation outperformed most of the newer algorithms. Only for the very lowest support settings on the bmspos dataset was it outperformed by any significant amount. The point is that for any significantly complex mining task, the transformation and mining phases will be applied offline. Whether an algorithm requires one versus two hours to complete is not a major concern if iteration is relegated to post-processing.

Dense datasets tell a different story. Most tabular datasets with more than a handful of columns are sufficiently dense to render minimum support pruning woefully inadequate. In the FIMI-03 experiments, minimum support was the only constraint considered, and the minimum support levels attainable by any algorithm on the densest datasets were nowhere near what would be necessary to find any reasonably predictive rules [6]. We must therefore ask, what other constraints might we employ? Another good constraint is that the mining artifacts, whether itemsets or rules, be in a sense non-redundant. In the rule mining context, I noted in [8] that when an itemset has support equivalent to that of one of its subsets, it is redundant in the sense that it leads only to rules that are equivalent to existing rules in predictive accuracy and the population covered. It is a simple matter to prune such itemsets in order to avoid excessive count-

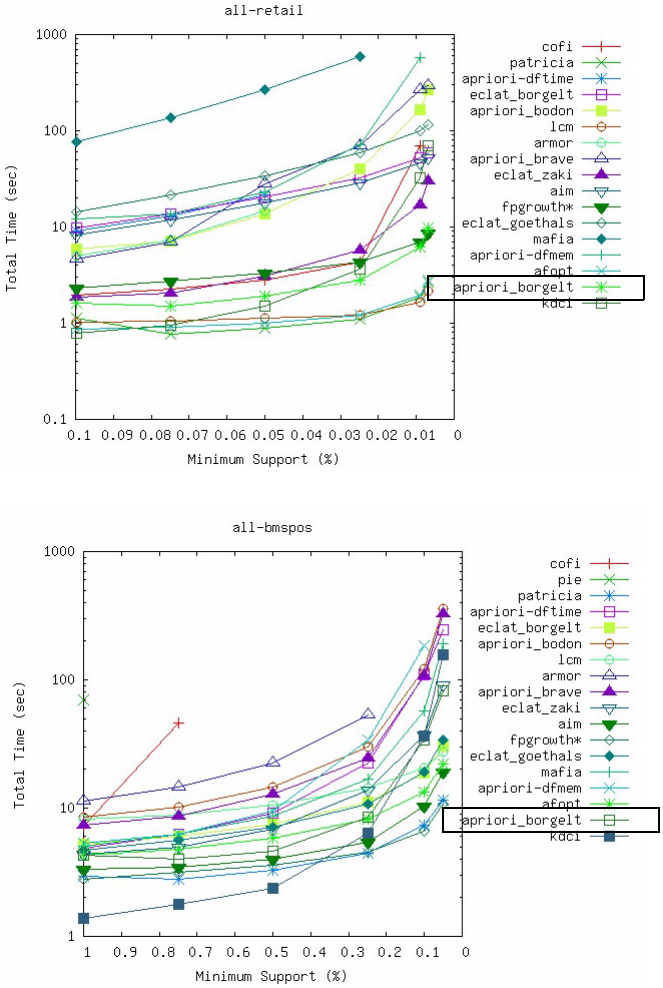


**Fig. 2.** Performance of the FIMI-03 implementations on sparse datasets

ing due to equivalent supports. This idea is the basis of what is now commonly known as freeness and closure [13,19,25] in the context of itemset mining, and also what I called “antecedent maximality” in the context of rule mining [6]. Closure, while not a parameterized property, is in a sense tweakable since we can easily regenerate non-closed itemsets (non-antecedent-maximal rules) from those in the set without dataset access. While it doesn’t always help significantly (e.g. for sparse datasets), it certainly doesn’t hurt performance. And finally, it doesn’t hinder discovery whatsoever since it removes only redundant information which can be efficiently derived if necessary.

Unfortunately, I feel the benefits of closure are often overstated. It’s no surprise that the Irvine “chess” and “connect-4” data-sets are the most common benchmarks for

demonstrating efficiency of closure-exploiting techniques. These datasets are relatively small and completely noise free. In the real world, data has noise. Noise quickly removes equivalence relations between itemsets, rendering closure-based pruning ineffective. Again, I refer to the FIMI-03 evaluation, where the algorithms employing closure remained ineffective on noisy dense data-sets such as pumsb, except under an unreasonably high minimum support constraint.

As I further noted in [8], it is therefore worth exploiting "near equivalence", which is when an itemset has a support value that is within a very small amount of one of its subsets. This idea has since been better formalized as the principle of $\delta$-free sets [13]. Pruning nearly equivalent itemsets restricts the questions we can ask from the result, but only slightly so. Further, it allows deriving of reasonably tight bounds on the support of any omitted itemset. While this makes the method very powerful in the itemset mining domain, these bounds cannot be straightforwardly used to intuitively quantify the effects on what rules are removed. But it is possible to perform some reasoning about rules using delta freeness, as demonstrated by Cremilleux and Boulicaut [14] who apply properties of delta free sets to characterizing classification rule conflicts. In the next section, I present what I believe to be a better constraint when rules instead of itemsets are the target pattern.

## 3 Discovery Preserving Rule-Specific Constraints

Rule-specific constraints are those that exploit properties specific to what distinguishes rules from itemsets; namely, the separation of the itemset into antecedent ($A$) and consequent ($C$) subsets (denoted $A \rightarrow C$). The most well known (and most often derided!) rule constraint is minimum confidence. Confidence, in the context of association rule mining, expresses the conditional probability with which the consequent holds given the antecedent:

$$\text{conf}(A \rightarrow C) = \frac{\text{sup}(A \cup C)}{\text{sup}(A)}$$

Confidence itself is not a bad metric. Along with knowledge of the background consequent probability (frequency), it conveys as much information as lift and related measures of predictive accuracy such as conviction [10]. It is my opinion that confidence, being a probability, is easier to interpret than these alternative measures. The problem with confidence stems from attempts at imposing a minimum bound in cases where the consequent of rules is allowed to vary, as is the case in the traditional association rule mining problem [1]. A single fixed minimum on confidence will exclude highly predictive rules if their consequents have a very low frequency, and will allow completely non-predictive rules if their consequents have high a high frequency. But for rules which share the same consequent, the fact is that confidence, lift (also known as interest) and conviction rank rules identically [7]. A minimum confidence constraint in the case of *consequent-constrained* rule mining is actually a very useful constraint, as it can be used to concisely exclude only non-predictive rules.[1] I believe that excluding non-predictive rules is discovery preserving in most contexts, provided "non-predictive" is quantified in a sufficiently tight manner.

A constraint that excludes non-predictive results is a good start, but the fact is we can do even better without unduly hindering discovery. Many rules are highly predictive, but when considered in the appropriate context, are actually of little interest if the goal is indeed to understand predictive relationships. For example, we might find a highly predictive rule $\{i_1, i_2\} \rightarrow \{i_c\}$. But what if the rule $\{i_1\} \rightarrow \{i_c\}$ is even more predictive? The rule $\{i_1, i_2\} \rightarrow \{i_c\}$ considered in isolation of such subrules might lead to highly suboptimal decisions. The point is that one cannot fully understand the predictive nature of a rule without also considering the predictive behavior of *all* its proper subrules. (Formally, a *subrule* of a rule $A \rightarrow C$ is any rule $A' \rightarrow C$ such that $A' \subseteq A$.) This idea extends the notion that interpreting a rule from its confidence without considering its consequent frequency is virtually meaningless.

So if we are to accept that the analyst is interested in discovering predictive relationships, another interesting and discovery preserving constraint would be to remove all rules containing subrules that are more (or equally) predictive. (Note that this notion encompasses pruning with support equivalence, since it's easy to show a rule that improves upon the predictive accuracy of all its subrules has no functional dependencies between disjoint subsets of its antecedent.) When applying this constraint in practice, the effects are indeed dramatic, but problems remain. While it is strictly more powerful than pruning with closure, we are still plagued by "near equivalence" relationships between an itemset and its subsets. These near equivalences result in numerous rule variations, each reflecting roughly the same relationships along with one or more "noise" items. For example, we may again have that $i_1$ strongly predicts $i_c$, say, with a confidence value of 90%. But we may also find that there exist dozens of other rules of the form $\{i_1\} \cup I \rightarrow \{i_c\}$ with confidence greater than 90% but perhaps less than 90.1%. Are these findings truly useful? I would argue in almost all cases they are not. One way to exclude such effects of near-equivalence is through a minimum positive bound on the predictive improvement a rule offers over all its subrules. Formally, I define the improvement value of a rule as the minimum of the differences between a rule's confidence and its proper subrules:

$$\text{improvement}(A \rightarrow C) = \min_{\forall A' \text{ s.t. } (A' \subset A)} \{\text{conf}(A \rightarrow C) - \text{conf}(A' \rightarrow C)\}$$

Note that alternate definitions of the improvement value are possible, for example we could have defined improvement using ratios between confidence or lift values instead of differences. I chose differences between confidences because I feel it is easier to interpret.

Instead of simply requiring that improvement be positive, we can instead require that the improvement value of any rule exceed a specified bound. For example, a minimum improvement bound of .1 would exclude the noise rules from the examples

---

1 A minimum confidence constraint also excludes negatively predictive rules, which are often of substantial interest. This can be avoided by also allowing the mining of rules that predict the negation of the desired consequent.

above. Experiments show that even such a very low minimum improvement setting dramatically reduces the size of the result set and the time required to compute it [7]. Much as delta-free sets allow tight bounds on the support of any omitted itemset, a minimum improvement filtered rule set allows for tight upper bounds on the predictive ability of any omitted rule.

## 4 Enforcing Rule Constraints During Mining

Rule constraints such as minimums on confidence, lift, conviction and improvement are not exploitable through generic constraint-based itemset mining frameworks [18] because they are not classifiable as monotone, anti-monotone, succinct, convertible, or by any other simple and easily exploitable property. How then can we hope to exploit them during mining? Let us first consider the confidence value which I now rewrite slightly:

$$\text{conf}(A \rightarrow C) = \frac{\text{sup}(A \cup C)}{\text{sup}(A \cup C) + \text{sup}(A \cup \neg c)}$$

In the above expression, the notation $\neg c$ reflects a conceptual item that is contained by any record that does not contain the consequent itemset. Such records are deemed "negative examples" in the machine-learning context. Rewritten as such, note that the expression is obviously monotone in $\text{sup}(A \cup C)$ and anti-monotone in $\text{sup}(A \cup \neg c)$. Given that confidence consists of monotone and anti-monotone constituents, is it possible to compute a reasonably tight bound on the confidence achievable during mining? The answer is yes. The key is to explicitly keep track of all items that can be appended to an itemset to form a descendent of the itemset in a tree-structured search space. In the description of the Max-Miner algorithm [4], I referred to such a structure as an itemset "group", though a more mathematically precise term is perhaps a "subalgebra" of the itemset lattice [9]. This concept has its roots in algorithms for circuit optimization [22], though Webb [24] was first to formalize the concept within a generic search framework, and also exploit it in rule mining tasks.

More formally, let's assume the consequent itemset is fixed to $C$ and we are searching over all possible antecedent itemsets for rules meeting various constraints. A node in the search space is represented by a head itemset $H$ representing the antecedent of the rule enumerated by the node, and another ordered itemset $T$ representing all items that can be appended to $H$ to form descendents of $H$ in the search space. At each node in the search space, before expanding the children of the node, we filter items from $T$ that cannot possibly lead to rules satisfying the constraints. In many cases, especially near the tree root, we may not be able to filter out any items. After filtering $T$, we obtain a new set $T'$ whose size dictates the number of children of the node. For each item $i$ in $T'$, the child expansion policy creates a new node with head set $H \cup \{i\}$ and tail set $\{j \mid j \in T' \wedge j \text{ follows } i \text{ in the item ordering}\}$.

Note then as we descend in the tree, the itemset $H$ always grows by exactly one item with each level (hence its support is monotonically decreasing), and the itemset $H \cup T$ either shrinks or stays the same (hence its support is monotonically increasing).

A bound on the confidence of any rule derivable by the node and its descendents can therefore be computed as follows:

$$\text{conf\_bound}(H \rightarrow C, T) = \frac{\sup(H \cup C)}{\sup(H \cup C) + \sup(H \cup T \cup \neg c)}$$

Correctness of the bound computation follows directly from the monotonicity and anti-monotonicity properties stated earlier. Conveniently, being able to bound confidence allows us to also bound improvement of a rule: we simply compute $\text{conf\_bound}(H \rightarrow C, T) - \text{conf}(H' \rightarrow C)$ for the proper subrule $H' \rightarrow C$ of $H \rightarrow C$ with the highest confidence.

This is a simple but illustrative example. In fact, we can break down the improvement value itself into monotone and anti-monotone constituents to more directly derive a complementary bound on the improvement attainable by any of a node's descendents. This example is considerably more involved, but the essential concepts are the same. For the details I direct the reader to [7]. While not immediately obvious, this particular bounding technique effectively exploits near equivalences between antecedent subsets.

Constraint enforcement is a search space size issue, but in the data mining literature, its presentation is often confusingly intertwined with the particular data management and traversal strategies employed. The constraint enforcement techniques such as those from above are generic, and can be applied irrespective of breadth versus depth-first search, and irrespective of whether we use database scans to compute supports compared to more esoteric dataset representations involving database projections or projected FP-tree structures. Contrary to what I sometimes find implied in the literature, using a depth-first search does not magically provide more pruning opportunities, though it may simplify and optimize the gathering of necessary support information to be able to apply them.

## 5 Conclusions

In summary, I have attempted to make several (hopefully controversial!) points regarding the hows, whys, and whens of constraints in itemset and rule discovery:

1. First and foremost, while the utility of constraints in knowledge discovery is undeniable, they should be applied judiciously. In particular, apply during mining only those constraints that are in a sense *discovery preserving*.
2. Apriori adequately solves the problem of mining itemsets and rules from market-basket and other sparse datasets.
3. Itemset freeness/closure/equivalence is a powerful concept, but its effectiveness in practice is limited. Consider exploiting constraints based on near-equivalence instead, such as $\delta$-freeness for itemsets and improvement thresholds for rules.
4. A rule cannot be fully interpreted in isolation from its subrules. This generalizes the well-known fact that confidence in absence of the consequent frequency is meaningless.

5. Itemset search methods should explicitly maintain the set of items that can be appended to the enumerated itemset in order to form its descendents in the search tree. Such explicit maintenance of lattice subalgebras allows exploiting both monotone and anti-monotone function constituents for more general and more powerful constraint enforcement.
6. Separating out issues of search space size from specific tree-traversal and data-management strategies improves understanding of algorithm performance, and increases the generality of constraint enforcement proposals.
7. Itemsets are not the only mining artifact of interest. Don't ignore the implications of mining-enforced constraints on what questions can be asked about the *rules*.

Now for the disclaimers. While I have stated each point as if it were maxim, I do not deny there are certain situations where some may fail to apply. Also, I do not claim to be the first to make them. Many similar points have been made within the literature (see for example [16] and [21] which are related point 1), and some I have picked up through osmosis from various talks and discussions. It is refreshing that even while I am writing this draft, I continue to come across new relevant work [3]. I must therefore apologize in advance for any references I have inevitably missed. For each point, I hope to minimally have provided some new perspectives.

## 6 References


1. R. Agrawal, T. Imielinski, and A. Swami. Mining Association Rules between Sets of Items in Large Databases. In *Proc. of the 1993 ACM-SIGMOD Conf. on Management of Data*, 207-216, 1993.
2. R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Fast Discovery of Association Rules. In *Advances in Knowledge Discovery and Data Mining*, AAAI Press, 307-328, 1996.
3. C. Antunes and A. L. Oliveira. Mining Patterns Using Relaxations of User Defined Constraints. In *Proc. of the Workshop on Knowledge Discovery in Inductive Databases*, 2004.
4. R. J. Bayardo. Efficiently Mining Long Patterns from Databases. In *Proc. of the 1998 ACM-SIGMOD Int'l Conf. on Management of Data*, 85-93, 1998.
5. R. J. Bayardo, The many roles of constraints in data mining. (Letter from the guest editor.) *ACM SIGKDD Explorations* 4(1), *i-ii*, June 2002.
6. R. J. Bayardo and R. Agrawal. Mining the most interesting rules. In *Proc. of the Fifth ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, 145-154, 1999.
7. R. J. Bayardo, R. Agrawal, and G. Gunopulos. Constraint-based rule mining in large, dense databases. In *Proc. of the 15th Int'l Conf. on Data Engineering*, 188-197, 1999.
8. R. J. Bayardo. Brute-force mining of high confidence classification rules. In *Proc. of the Third International Conference on Knowledge Discovery and Data Mining*, 123-126, 1997.
9. C. Bucila, J. Gehrke, D. Kifer, DualMiner: A Dual-Pruning Algorithm for Itemsets with Constraints. In *Proc. SIGKDD-2002*. In Proc. SIGKDD 2002.
10. S. Brin, R. Motwani, J. Ullman, and S. Tsur. Dynamic Itemset Counting and Implication Rules for Market Basket Data. In *Proc. of the 1997 ACM-SIGMOD Conf. on Management of Data*, 255-264, 1997.

11. R. J. Brachman and T. Anand. The Process Of Knowledge Discovery In Databases: A Human-Centered Approach. In *Advances In Knowledge Discovery And Data Mining*, eds. U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, AAAI Press/The MIT Press, Menlo Park, CA., 37-57, 1996.
12. B. Goethals and M. J. Zaki. Advances in Frequent Itemset Mining Implementations: Introduction to FIMI-03. In *Proc. of the ICDM 2003 Workshop on Frequent Itemset Mining Implementations*, 2003.
13. J.-F. Boulicaut, A. Bykowski, and C. Rigotti. Approximation of frequency queries by means of free-sets. In *Proc. PKDD Int. Conf. Principles of Data Mining and Knowledge Discovery*, pages 75-85, 2000.
14. B. Cremilleux, J-F. Boulicaut. Simplest rules characterizing classes generated by delta-free sets. In: *Proceedings of the 22nd BCS SGAI International Conference on Knowledge Based Systems and Applied Artificial Intelligence*, Cambridge (UK), 33-46, 2002.
15. B. Jeudy and J.-F. Boulicaut. Using Condensed Representations for Interactive Association Rule Mining. In *Proc. of Principles of Data Mining and Knowledge Discovery: 6th European Conference (PKDD 2002)*, 228-236, 2002.
16. J. Hipp and U. Güntzer. Is pushing constraints deeply into the mining algorithms really what we want?: an alternative approach for association rule mining. *ACM SIGKDD Explorations* 4(1), 50-55, June 2002.
17. B. Nag, P. M. Deshpande, and D. J. DeWitt. Using a knowledge cache for interactive discovery of association rules. In *Proc. SIGKDD-1999*, 244-253, 1999.
18. R. Ng, L. V. S. Lakshmanan, J. Han, and A. Pang. Exploratory mining and pruning optimizations of constrained associations rules. In *Proc. SIGMOD-1998, 13-24, 1998.*
19. N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Efficient mining of association rules using closed itemset lattices. *Information Systems*, 24(1):25-46, 1999.
20. R. Rymon. Search through systematic set enumeration. In *Proc. of the Third Int'l Conf. on Principles of Knowledge Representation and Reasoning*, 539-550, 1992.
21. S. Sahar: Interestingness via What is Not Interesting. In *Proc. of SIGKDD-1999*: 332-336
22. J. R. Slagel, C.-L. Chang, and R. C. T. Lee. A New Algorithm for Generating Prime Implicants. *IEEE Trans. on Computers*, C-19(4):304-310, 1970.
23. R. Srikant, Q. Vu, and R. Agrawal. Mining Association Rules with Item Constraints. In *Proc. of the Third Int'l Conf. on Knowledge Discovery in Databases and Data Mining*, 67-73, 1997.
24. G. I. Webb. Opus: an efficient admissible algorithm for unordered search. *Journal of Artificial Intelligence Research* 3, 431-465, 1995.
25. M. J. Zaki. Generating non-redundant association rules. In *Proc. SIGKDD-2000*, pages 34-43, 2000.