Just Enough Learning (of Association Rules)* †

Tim Menzies
Lane Department of Computer Science
University of West Virginia
PO Box 6109 Morgantown
WV 26506-6109, USA
tim@menzies.com

Ying Hu
Dept. Electical & Computer Engineering
University of British Columbia
2356 Main Mall; Vancouver
B.C. Canada V6T1Z4.
yingh@ece.ubc.ca

ABSTRACT

An over-zealous machine learner can automatically generate large, intricate, theories which can be hard to understand. However, such intricate learning is not necessary in domains that lack complex relationships. A much simpler learner can suffice in domains with *narrow funnels*; i.e. where most domain variables are controlled by a very small subset.

Such a learner is TAR2: a weighted-class minimal contrast-set association rule learner that utilizes confidence-based pruning, but not support-based pruning. TAR2 learns *treatments*; i.e. constraints that can change an agent's environment. Treatments take two forms. *Controller treatments* hold the *smallest number* of conjunctions that *most improve* the current state of the system. *Monitor treatments* hold the *smallest number* of conjunctions that best detect future faulty system behavior. Such treatments tell an agent what to do (apply the controller) and what to watch for (the monitor conditions) within the current environment.

Because TAR2 generates very small theories, our experience has been that users prefer its tiny treatments. The success of such a simple learner suggests that many domains lack complex relationships.

Keywords

Association rules, treatment learning, contrast sets, ML-lite.

1. INTRODUCTION

When is just enough learning enough? If we replace sophisticated and complex learners by simpler algorithms, what do we lose? Do such simpler learners have any generality? Or are they just one-off ad-hoc hacks that must be discarded whenever the domain changes?

These kind of questions keep repeating, in many fields. For example, proponents of lightweight case-based reasoning (*CBR-lite*) reject intricate domain modelling, arguing that much can be achieved using (e.g.) very simple indexing schemes[1]. Similarly, in the *testing-lite* field, Menzies & Cukic have argued that a surprisingly large percentage of errors can be detected within software via an inexpensive random sampling of the input space of a program. Analogous results exist in the machine learning (ML) community. For example, Holte found that C4.5 performs only moderately better than 1R (1R is a learner which only generates decision trees with a maximum depth of one) [11]

Lightweight approaches may not necessarily replace heavier methods. For example, the *CBR-heavy* community rejects *CBR-lite*, arguing that *CBR-lite* tends to dodge the most interesting and fundamental problems. Menzies & Cukic caution that *testing-lite* is inappropriate for mission-critical and safety-critical systems. Similarly, Holte does not propose 1R as a replacement for C4.5; rather, he used it to argue that most of the datasets used to test learners does not contain very complex relationships. When ML-lite methods such as 1R miss such complex relationships, heavier weight methods are required.

On the other hand, if most domains lack such complex relationships, then ML-lite is an attractive option. One advantage of exploring simpler methods before complex ones is that ML-lite algorithms can generate very simple theories. Such simpler theories are easier to understand.

We have reasons to believe that many domains lack complex relationship. For some years now, we have repeatedly observed a curious *narrow funnel effect*. In many domains, it has been observed that a small number of critical variables control the remaining variables within a system, the metaphor being that all processing runs down the same narrow funnel [25]. The concept of narrow funnels has been reported in many domains under a variety of names including:

- *Master-variables* in scheduling [7];
- *Prime-implicants* in model-based diagnosis [28] or machine learning [29], or fault-tree analysis [14].
- *Backbones* in satisfiability [26, 30];
- *the dominance filtering* used in Pareto optimization of designs [12];
- *Minimal environments* in the ATMS [8];
- The *base controversial assumptions* of HT4 [24].

*KDD paper ID: 440

†Submitted to the KDD-2002, The Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23 - 26, 2002, Edmonton, Alberta, Canada `http://www.acm.org/sigkdd/kdd2002/`. Wp ref 01/tar2/tar2

[1]European Case-Based Reasoning Workshop, Lausanne Switzerland, 1996, `http://www.ai-cbr.org/ewcbr96.html`.

Whatever the name, the core intuition in all these terms is the same: what happens in the total space of a system can be controlled by a small critical region. Where the narrow funnel effect exists, the space of options within a large space reduces to just the range of a few variables within the narrow funnel. In such a reduced space, variables assignments outside the funnel are highly correlated to assignments within the funnel. Machine learning in such domains is very simple: an adequate theory need only comment on assignments to the variables that are highly correlated to funnel assignments.

Note that Holte's results are consistent with ML data sets he explored contained narrow funnels. 1-level deep decision trees work since that is enough to check for assignments to the funnels variables.

We have argued previously that narrow funnels are very common [16, 17, 21, 23]. Hence, ML-lite tools should suffice in many domains. This paper tests that hypothesis. Since Holte has already studied ML-lite for decision tree learning, this paper explores TAR2: a lightweight association rule learner. Our premise will be the *small treatment assumption*; i.e. a very small number of associations are adequate for controlling domains with narrow funnels. Our associations take the form of *treatments*; i.e. a constraint that can change an agent's environment. Treatments take two forms. *Controller treatments* hold the *smallest number* of conjunctions that *most improve* the current state of the system. *Monitor treatment* hold the *smallest number* of conjunctions that best detect future faulty system behavior. Such treatments tell an agent what to do (apply the controller) and what to watch for (the monitor conditions) within the current environment.

TAR2's distinguishing feature is that it performs very well, yet it is seems overly simplistic. The algorithm outputs only two associations: the best smallest controller and the best smallest monitor. If domains contain complex relationships, then these two small associations will be useless. The algorithm's runtimes are exponential on the size of the treatments. Unless the small treatment assumption holds, such exponential runtimes can be impractically slow. Also, the algorithm relies on a *confidence1* measure which prunes the space of possible associations. The confidence1 measure we describe below has no special merit: it was merely the first one we could think of. Further, our initial implementation worked without algorithmic or memory management optimizations. Our only explanation for the surprising success of this simplistic implementation is that the small treatment assumption holds for the domains we studied.

Methodologically, proponents of a lightweight method are duty-bound to describe an operational assessment criteria. Our criteria is based on treatments: a good association selects a portion of the test data that contains something we desire or something we with to avoid. Without such a criteria, it is hard to assess lightweight vs heavyweight approaches. In our view, such a criteria must be *algorithmic*, *semantic* and *portable*. Reports of the effectiveness of an algorithm such as learner L could find associations within 10,000,000 examples using only 843MB [2] are exciting. However, the *algorithmic* nature of this report does not assess any *semantic* issues. If the learnt associations weren't "useful", however that is defined, then that learning effort was wasted. "Usefulness" can be assessed via domain experts who understand the semantics of a domain. Such a measure is not *portable*: researchers across the globe rarely have access to the same domain experts.

The advantage of treatment-based assessment is that they are a portable semantic assessment criteria. That is, other researchers could demonstrate the "usefulness" of their learner on the examples shown in this paper, without having to (e.g.) interview domain experts. The disadvantage of treatment-based assessment is that it is not a widely-accepted criteria. Other papers in the fields of association rule learner do not express the efficacy of their associations in terms of the treatments found in their associations. Hence, this paper can only offer *baseline measurements*, not comparative measurements, of a new sub-class of learner; i.e. a treatment association rule learner.

The rest of this article discusses TAR2. After an introductory example and a discussion of related work, the TAR2 algorithm is presented. This is followed by examples and evaluations and an analysis of the general applicability of our approach.

2. RELATED WORK

Formally, TAR2 is a *weighted-class minimal contrast-set* association rule learner that uses *confidence measures* but not *support-based pruning*. This section discusses those terms.

The general form of a treatment is:

$$\begin{array}{lll} R_1 & if & Attr_1 = range_1 \wedge Attr_2 = range_2 \wedge ... \\ & then & good = more \wedge bad = less \\ R_2 & if & Attr_1 = range_1 \wedge Attr_2 = range_2 \wedge ... \\ & then & good = less \wedge bad = more \end{array}$$

where R_1 is the controller rule; R_2 is the monitor rule; *good* and *bad* are sets of classes that the agent likes and dislikes respectively; and *more* and *less* are the frequency of these classes, compared against the current situation, which we call the *baseline*. The nature of these output rules distinguishes TAR2 from many other learning strategies.

Association rule learning: Classifiers like C4.5 and CART learn rules with a single attribute pair on the right-hand side; e.g. *class= goodHouse*. Association rule learners like APRIORI [2] and TAR2 generate rules containing multiple attribute pairs on both the left-hand-side and the right-hand-side of the rules. That is, classifiers have a small number of pre-defined targets (the classes) while, for association rule learners, the target is less constrained.

General association rule learners like APRIORI input a set of D transactions of items I and return associations between items of the form $LHS \Rightarrow RHS$ where $LHS \subset I$ and $RHS \subset I$ and $LHS \cap RHS = \emptyset$. A common restriction with classifiers is that they assume the entire example set can fit into RAM. Learners like APRIORI are designed for data sets that need not reside in main memory. For example, Agrawal and Srikant report experiments with association rule learning using very large data sets with 10,000,000 examples and size 843MB [2]. However, just like Webb [32], TAR2 makes the "memory-is-cheap assumption"; i.e. TAR2 loads all it's examples into RAM.

Specialized association rule learners like CBA [13] and TAR2 impose restrictions on the right-hand-side. For example, TAR2's right-hand-sides show a prediction of the *change* in the class distribution if the constraint in the left-hand-side were applied. The CBA learner finds *class association rules*; i.e. association rules where the conclusion is restricted to one classification class attribute. That is, CBA acts like a classifier, but can process larger datasets that (e.g.) C4.5. TAR2 restricts the right-hand-side attributes to just those containing criteria assessment.

Weighted-learning: Standard classifier algorithms such as C4.5 [27] or CART [4] have no concept of class *weighting*. That is, these systems have no notion of a *good* or *bad* class. Such learners therefore can't filter their learnt theories to emphasize the location of the *good* classes or *bad* classes. Association rule learners such as MINWAL [5], TARZAN [22] and TAR2 explore *weighted learning* in which some items are given a higher priority weighting that

others. Such weights can focus the learning onto issues that are of particular interest to some audience. For example TARZAN [22] swung through the decision trees generated by C4.5 [27] and 10-way cross-validation. TARZAN returned the smallest treatments that occurred in most of the ensemble that *increased* the percentage of branches leading to some preferred highly weighted classes and *decreased* the percentage of branches leading to lower weighted class. TAR2 was as experiment with applying TARZAN's tree pruning strategies directly to the C4.5 example sets. The resulting system is simpler, fast to execute, and does not require calling a learner such as C4.5 as a sub-routine.

Contrast sets: Instead of finding rules that describe the current situation, association rule learners like STUCCO [3] finds rules that differ meaningfully in their distribution across groups. For example, in STUCCO, an analyst could ask "what are the differences between people with Ph.D. and bachelor degrees?". TAR2's variant on the STUCCO strategy is to combine contrast sets with weighted classes with minimality. That is, TAR2 treatments can be viewed as the smallest possible contrast sets that distinguish situations with numerous highly-weighted classes from situations that contain more lowly-weighted classes.

Support-based pruning: In the terminology of APRIORI, an association rule has *support* s if $s\%$ of the D contains $X \wedge Y$; i.e. $s = \frac{|X \wedge Y|}{|D|}$ (where $|X \wedge Y|$ denotes the number of examples containing both X and Y). The *confidence* c of an association rule is the percent of transactions containing X which also contain Y; i.e. $c = \frac{|X \wedge Y|}{|X|}$.

Many association rule learners use *support-based pruning* i.e. when searching for rules with high confidence, sets of items $I_i, ...I_k$ are only be examined only if all its subsets are above some minimum support value. Support-based pruning is impossible in weighted association rule learning since with weighted items, it is not always true that subsets of *interesting* items (i.e. where the weights are high) are also interesting [5]. Another reason to reject support-based pruning is that it can force the learner to only miss features that apply to a small, but interesting subset of the examples [31].

Confidence-based pruning: Without support-based pruning, association rule learners rely on confidence-based pruning to reject all rules that fall below a minimal threshold of adequate confidence. TAR2 uses *confidence1* pruning.

3. CONFIDENCE1 PRUNING

TAR2 targets the attribute ranges that "nudge" a system away from undesired behavior and towards desired behavior. TAR2's score for each range is the *confidence1* measure. This value is high if a range occurs frequently in desired situations and infrequently in undesired situations. That is, if we were to impose this range as a constraint, then it would tend to "nudge" the system into better behavior.

To find confidence1, we assume that we can access $\$class$; i.e. some numeric value assigned to $class$. The class with the highest value is the $best$ class. The $lesser$ classes are the set of all classes, less the $best$ class. Let $O[C, A.R]$ be the number of occurrences of some attribute range in some class C; i.e.

$$O[C]_{A.R} = |A.R \wedge class = C \wedge D|$$

To generate confidence1, we compare the relative frequencies of an attribute range in different classes. This comparison is weighted by the difference in the scores of the classes, and normalized by the total frequency count of the attribute range; i.e.

Items				*Criteria*
outlook	*temp(^oF)*	*humidity*	*windy?*	*class*
sunny	*85*	*86*	*false*	*none*
sunny	*80*	*90*	*true*	*none*
sunny	*72*	*95*	*false*	*none*
rain	*65*	*70*	*true*	*none*
rain	*71*	*96*	*true*	*none*
rain	*70*	*96*	*false*	*some*
rain	*68*	*80*	*false*	*some*
rain	*75*	*80*	*false*	*some*
sunny	*69*	*70*	*false*	*lots*
sunny	*75*	*70*	*true*	*lots*
overcast	*83*	*88*	*false*	*lots*
overcast	*64*	*65*	*true*	*lots*
overcast	*72*	*90*	*true*	*lots*
overcast	*81*	*75*	*false*	*lots*

Figure 1: A log of some golf-playing behavior.

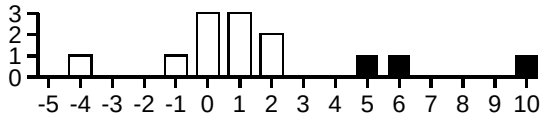


Figure 2: Frequency of confidence1 generated from Figure 1. Assumes that numeric ranges have been divided into 3 $bands$. Outstandingly high confidence1 values shown are in black. Y-axis is the number of ranges that have a particular confidence1 value.

$$\frac{\sum_{C \in lesser} ((\$best - \$C) * (O[best]_{A.R} - O[C]_{A.R}))}{|A.R \wedge D|}$$

For example, from the golf playing example of Figure 1, let us assume that the classes have been scored as follows: "lots"=8, "some"=4, "none"=2; i.e. "lots" is the $best$ class. The range *outlook=overcast* appears four, zero, and zero times when playing "lots", "some", and "none" golf (respectively). The confidence1 of *outlook=overcast* is therefore:

$$\frac{((8-2)*(4-0)) + ((8-4)*(4-0))}{4+0+0} = 10$$

Figure 2 shows the range of confidence1 seen in Figure 1. The confidence1 ranges shown in black are outstandingly high; i.e. these are the values may generate the best control treatments. TAR2 forms its treatments by exploring subsets of the ranges with outstandingly high confidence1 values.

4. INSIDE TAR2

TAR2 generates controller and monitor treatments. Monitors are generated using in same manner as generating controllers. However, before the monitor is generated, the scoring function for the criteria is reversed so TAR2 now seeks attribute ranges that *nudge* a system into worse behavior. The rest of this section discusses how to generate controllers.

The TAR2 algorithm is shown in Figure 3. The `frequency` function counts the frequency of examples falling into different criteria. Using this function, a $baseline$ class distribution is collected from D (this is used later to contrast different treatments) and copied to a $temp$ variable (this is used to store the best distribution seen so far). The `compare` function compares two frequencies to generate reports like (e.g.) 43% less "lots" and 5% less "some" and 167% more "none". The `discretize` function

input:	D	The examples.
	$items$	Attributes seen in the examples.
	$best$	The best combination of criteria.
	N	Desired size of LHS.
	$promising$	Threshold for a useful attribute range.
	$skew$	Threshold for acceptable number of $best$ entries in $treated$.
	$bands$	Number of divisions within continuous ranges.
output:	lhs	A conjunction of attribute ranges
	rhs	a change in the class distributions

```
01. D1 ← discretize(D, bands)
02. temp ← baseline ← frequency(D1)
03. for attribute in items {
04.     for R in attribute.ranges {
05.         if   confidence1(attribute.R) ≥ promising
06.         then candidates ← candidates + attribute.R}}
07. for C ⊆ candidates where |C| = N {
08.     treated ← C ∧ D1
09.     result ← frequency(treated)
10.     if   result>temp and |best∧D1|/|best∧treated|>skew
11.     then    {lhs ← C
12.              rhs ← compare(baseline,result)
13.              temp ← result}}
14. if (lhs ≠ ∅ and rhs ≠ ∅) then return (lhs, rhs)
15. else return "no treatment"
```

Figure 3: The TAR2 algorithm.

```
controllerG
if   outlook=overcast
then (230% more "lots" and no "some"
     and no "none").

monitorG
if    90 <= humidity < 97
then (43% less "lots" and 5% less "some"
     and 167% more "none").
```

Figure 4: Control and monitor rules found from Figure 1. To control *outlook*, unscrupulous owners of golf courses could (e.g.) bribe radio announces to lie about the weather report.

divides the numeric ranges seen in the examples into $bands$ number of groups. Numerous discretization policies are possible but the simplest works adequately: TAR2 sorts the known values and divides into $bands$with (roughly) the same cardinality.

Once a treatment is found, it is applied to the example set to create a $treated$ example set; i.e. all the examples that don't contradict the proposed treatment (see line 8). A "good" treatment includes most of the examples that have the $best$ criteria (e.g. in the golf example of Figure 1, $best$= playing "lots" of golf). The $skew$parameter is used at line 10 to reject "bad" treatments; i.e. those that don't contain enough of the $best$ criteria. For example, at $skew$=5, at least 20% of the $best$ criteria must appear in the treatment.

TAR2 explores subsets of the $ranges$ found in a set of examples D (see line 7). Subset exploration is constrained to just the $ranges$ with an outstandingly large confidence1 score (see line 5). Even with this restriction, there are still an exponential number of such subsets. Hence, to be practical, TAR2 must seek the *minimal* possible number of control actions and monitors. Accordingly, the user of TAR2 constrains its learning to rule conditions of size N, where N is small (see line 7). Often, effective treatments can be found using $N \leq 4$ which suggests that narrow funnels existed in the datasets used for our case studies.

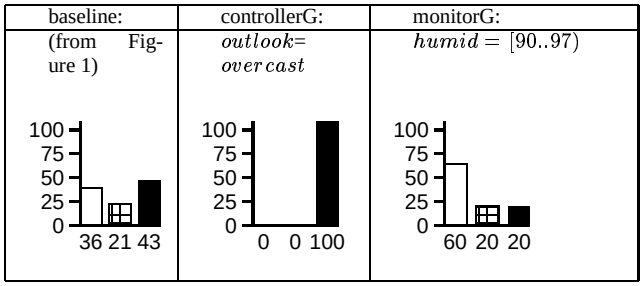


Figure 5: Percentage of classes seen in different situations. The left-hand-side histogram is a report of the class frequencies seen in Figure 1. The middle and right-hand-side histograms were generated by applying the treatments of Figure 4. KEY: none; some; lots.

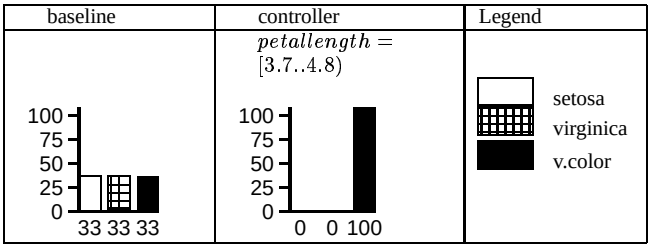


Figure 7: Iris

5. EXAMPLES AND EXPERIMENTS

5.1 Examples

The output of TAR2 describes constraints which, if applied to the dataset, may reject certain examples. For example, the `controllerG` treatment of Figure 4 contains the constraint $outlook = overcast$. If we reject all items in the golf dataset that contradicts this constraint, then our golfers now play "lots", "some", and "none" golf in 100%, 0%, and 0% (respectively) of the constrained dataset (as shown in the middle histogram of Figure 5).

The monitor rule `monitorG` of Figure 4 was generated in a similar manner; but with the scoring system reversed; i.e. "lots"=2, "some"=4, "none"=8. In this case, "none" is the "$best$" class and TAR2 will find a treatment that selects for less golf behavior; i.e. $90 \leq humidity < 97$. After applying this constraint , the class distribution changes to the right-hand-side histogram of Figure 5.

Figure 6 contrasts two theories learnt from the same data set using TAR2 and C4.5. Note that TAR2's theory is far smaller than C4.5's theory. Our experience with business users is that they prefer find TAR2's simpler theories. As one user put it "decision trees just tell you where you are, treatments tell you what to do".

5.2 Experiments

This section discusses experiments with TAR2 where the leaner was assessed via two methods:

Xvals: Standard N-way cross-validation studies.

Simulations: Simulations showing how well TAR2's treatments can control or monitor some model.

Figure 6.A: A learnt decision tree. Classes (right-hand-side), top-to-bottom, are "high", "medhigh", "medlow", and "low" This indicates median value of owner-occupied homes in $1000's.

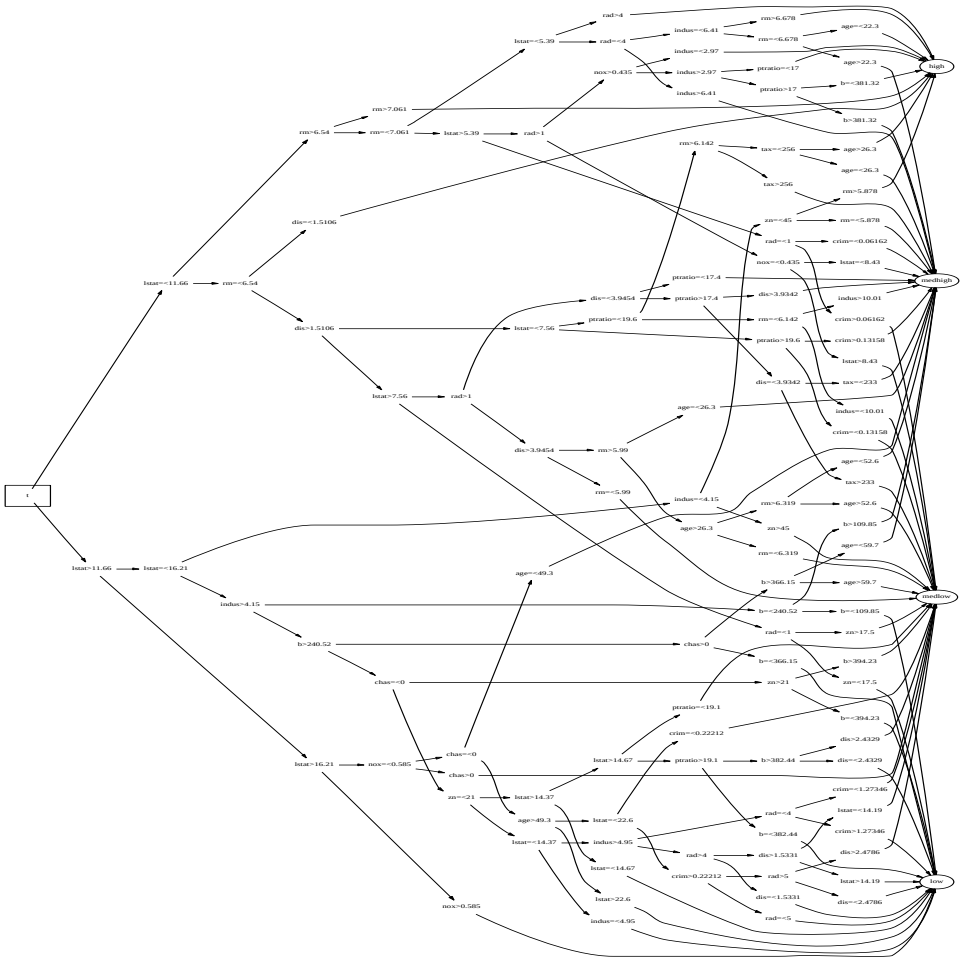


Figure 6.B: Treatments learnt in the same domain:

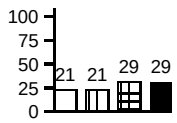
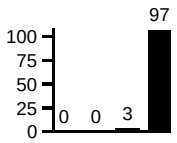
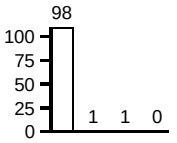
	baseline	`controllerH` (i.e. best action)	`monitorH` (i.e. diaster if..)
Treatment	nothing	$6.7 \leq RM < 9.8$ $\wedge 12.6 \leq PTRATION < 15.9$	$0.6 \leq NOX < 1.9$ $\wedge 17.16 \leq LSTAT < 39.0$
Results			
N	506	38	81

Figure 6: Comparing decision trees learnt by C4.5 and treatments learnt by the TAR2 treatment learner. Theories learnt from the 506 cases in HOUSING example set from the UC Irvine repository. This dataset has the class distribution shown in the bottom table, left-hand-side. The bottom table shows actions that should most increase/decrease housing values in the middle/right columns (respectively). LSTAT= lower status of the population; NOX= nitric oxides concentration (parts per 10 million); PTRATIO= pupil-teacher ratio by town; RM= average number of rooms per dwelling.

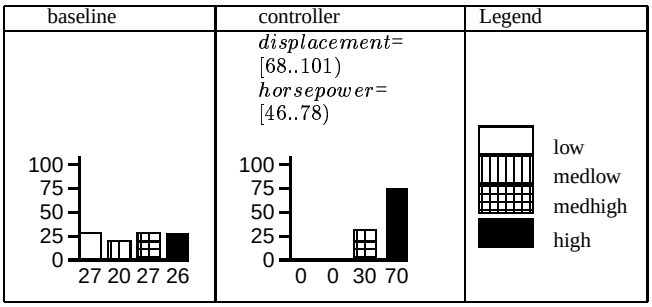


Figure 8: Autompg

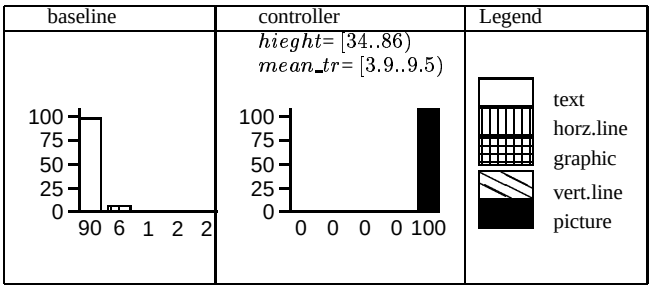


Figure 9: Page-block

5.2.1 Xval Studies

Figure 7 to Figure 11 shows TAR2 executing over some samples from the UC Irvine repository (`http://www.ics.uci.edu/~mlearn/`):

- These figures display the effects of the treatment closest to the average improvement seen in a 10-way cross-validation study.
- These figures show the class distributions in percentages
- These figures list the domain classes in the a legend. The heuristic worth assigned to each class is, top-to-bottom, worst-to-best.

In Figure 7, TAR2 was told that the worth of each type of flower was (in increasing order) *setosa*, *viginica*, then *v.color*. TAR2 then learnt that $3.7 \leq petallength < 4.8$ would select for the flower with highest worth (i.e. *v.color*).

Similarly, in Figure 8, TAR2 learnt a selector that favored high quality cars. By restricting engine size to $68 \leq displacement < 101$ and $46 \leq horsePower < 78$, the ratio of high quality cars

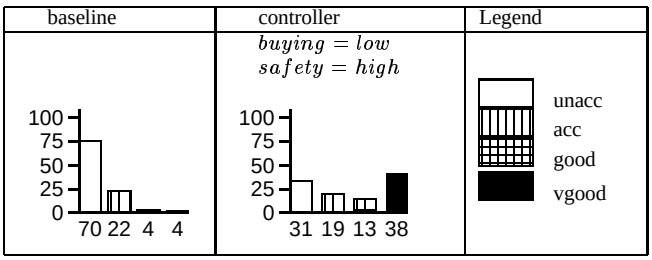


Figure 10: Car

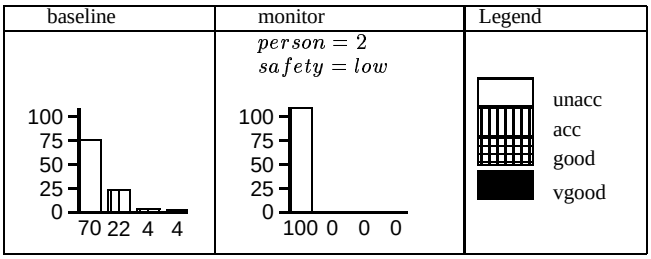


Figure 11: Car: reversing the class scoring.

increased from 26% to 70%. Further, the low and medium low cars have disappeared.

In Figure 9, TAR2 learnt a specialized feature extractor for finding pictures mixed in with text, horizontal lines, vertical lines, and graphics. According to TAR2, a height between 34 to 86, and a mean number of white-black transitions between 3.9 and 9.5 will locate text blocks, and nothing else.

In the car domain of Figure 10, most of the classes are non-best. The average best controller seen in the 10-way cross-validation for the car domain was *buying=low* and *safety=high*. While this controller increases the frequency of very good cars from 4% to 38%, this controller still leaves us with 31% unacceptable cars. While this controller is weak, the monitor obtained by reversing the class scoring is very strong. Figure 11 shows that monitor: if we select two person cars with low safety, then 100% of the cars are unacceptable. That is, when the best class occurs rarely in the dataset, TAR2 may be better at finding methods to *degrade* a system, rather than *improve* it.

5.2.2 Simulation Studies

Another way to assess TAR2 is to test how well it can control some model. To perform such an assessment, we (i) generated data sets from some model; (ii) apply TAR2 to find treatments from those data set; (iii) imposed those treatments as constraints on the model; (iv) ran the model a second time; (v) compared the outputs of the second run to the predictions made by TAR2.

In our first two simulations studies, a *baseline* class distribution was used by TAR2 to generate a best controller and a prediction of how this best controller would change the class distribution. We call the predicted distribution the *treated* distribution. The *actual* distribution was the class distribution seen after the best controller was imposed on the model and the model executed again. In Figure 12 and Figure 13, the treated distribution matches the result distribution almost exactly; i.e. TAR2 accurately predicted the effects of the controller treatment.

Figure 12 was generated from a model of software project risk. This risk model was implemented as part of the COCOMO project. The goal of the COCOMO project is to build an open-source software cost estimation model [1]. Internally, the model contains a matrix of parameters that should be tuned to a particular software organization. Using COCOMO-II, the Madachy risk model can assess the risk of a software cost over-run [15]. For machine learning purposes, the goal of using the Madachy model is to find a change to a description of a software project that reduces the likelihood of a poor risk software project [19, 22]. In the experiment shown in Figure 12, the model was executed 30,000 times using randomly selected inputs. When the treatments learnt from TAR2 treatments were imposed on model inputs, and the model was executed again, all the high risk projects were removed, the percentage of medium risk projects was significantly reduced, and the percentage of low

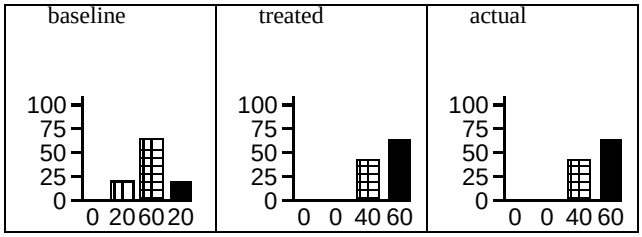


Figure 12: COCOMO key: very high risk; high risk; medium risk; low risk.

risk projects was tripled.

Figure 13 shows TAR2 controlling a qualitative description of an electrical circuit. A qualitative description of a circuit of 47 wires connecting 9 light bulbs and 16 other components was coded in Prolog. The model was expressed as a set of constraints; e.g. the `sum` of the voltages of components in series is the `sum` of the voltage drop across each component. The goal of the circuit was to illuminate a space using the 9 light bulbs. The circuit is qualitative and qualitative mathematics is nondeterministic; e.g. sum of a negative and a positive value is unknown. The problem with the circuit was out-of-control nondeterminism. On backtracking, this circuit generated 35,228 different solutions to the constraints. In many of these solutions, the circuit was unacceptably dark: only two bulbs glowing, on average (see the top histogram of Figure 13) . The goal of the machine learning was hence to find a minimal set of changes to the circuit to increase the illumination [18]. Figure 13 shows the distribution of the frequency with which bulbs glowed in a qualitative circuit description. The behavior of qualitative circuits is notoriously hard to predict [6] but TAR2 found two actions on the circuit that trebled the average number of bulbs that glowed (see the *treated* and *actual* plot of Figure 13).

Figure 14 shows a third simulation study with TAR2. Analysts at the NASA Jet Propulsion Laboratory debate satellite design by building a semantic network connecting design decisions to satellite requirements [9]. Each edge is annotated with the numeric cost and benefits of taking some action. Some of these nodes represent base decisions within the project (e.g. selection of a particular type of power supply). Each set of decisions has an associated cost. The net can be executed by selecting actions and seeing what benefits results. One such network included 90 possible actions; i.e. $2^{99} \approx 10^{30}$ combinations of actions. Note the black line, top-left, of Figure 14. All the dots below this line were generated via 10,000 random selections of the decisions, and the collection

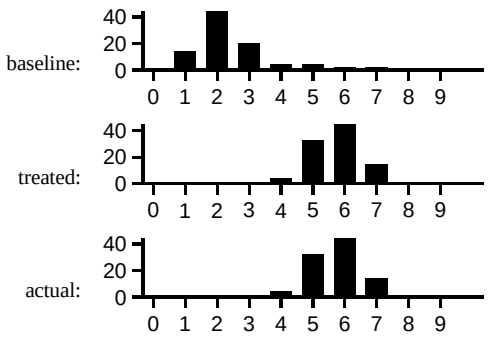


Figure 13: Circuit. X-axis denotes number of bulbs glowing in the circuit.

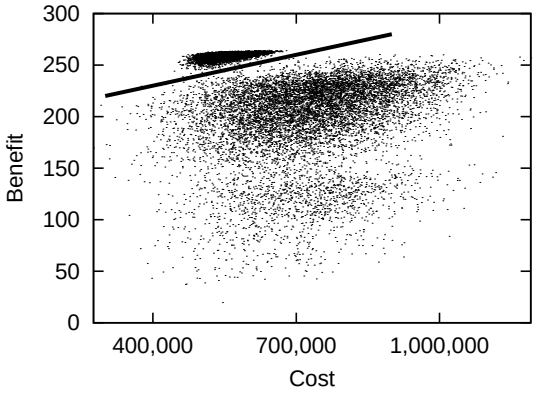


Figure 14: Results from the satellite domain. The dots below the line show the initial output of the model: note the very large spread in the costs and benefits. The dots above the line show the final outputs of the model after 5 iterations of TAR2 learning.

of their associated costs and benefits. All the dots above this line represent high benefit, low cost projects found by TAR2 [10]. In this application, TAR2 was used as a knowledge acquisition tool. After each run of TAR2, the proposed best controller was debated with the analysts. Each run, and its associated debate, resulted in a new set of constraints for the semantic net. The new constraints were then imposed on the model before the next run. After five runs, TAR2 found 30 decisions (out of 99) that crucially effected the cost/benefit of the satellite. Note that this means TAR2 also found 99-30=67 decisions that could be safely ignored.

For comparison purposes, a genetic algorithm (GA) was also applied to the Figure 14 domain [10]. The GA also found decisions that generated high benefit, low cost projects. However, each such GA solution commented on every possible decisions and there was no apparent way to ascertain which of these are the most critical decisions. The TAR2 solution was deemed superior to the GA solution by the domain experts, since the TAR2 solution required just 30 actions rather than the 99 demanded by the GA.

Note that the Figure 14 case study is not a counter example to our thesis that most domains have narrow funnels. That study adopted the incremental approach for reasons of convenience. JPL's semantic net simulator was too slow to generate enough examples at one run. Hence, an incremental generate-and-constrain approach was taken.

6. GENERALITY

This section is an algorithmic assessment of TAR2. Such an algorithm assessment comments on TAR2's ability to scale to larger domains.

Figure 15 reports TAR2 runtimes on data sets of different sizes. Figure 16 shows three studies where the size of the treatments (N, from line 7 in Figure 3) was held constant, and the size of the dataset was increased. Figure 17 shows one study were the size of the dataset was held constant and the size of the treatments was increased. Note that:

1. TAR2 can handle small to medium sized datasets. For example, the algorithm learnt effective treatments in 23 seconds from a dataset containing size 250,000 examples: see the *reachness* domain in Figure 15.

domain	# examples	Attributes # continuous	Attributes # discrete	# classes	treatment size	runtimes (secs)
From UC Irvine:						
iris	150	4	0	3	1	<1
wine	178	13	0	3	2	<1
car	1,728	0	6	4	2	<1
autompg	398	6	1	4	2	1
housing	506	13	0	4	2	1
page blocks	5,473	10	0	5	2	2
From this article:						
circuit	35,228	0	18	10	4	4
cocomo	30,000	0	23	4	1	2
satellite	30,000	0	99	9	5	86
From other sources [20]:						
reachness	25,000	4	9	4	2	3
	250,000	4	9	4	1	23

Figure 15: Runtimes for TAR2 on different domains (on a 333MHz Windows machine with 200MB of ram). The text discusses experiments with 10,000 examples from the satellite domain. This table shows a larger case study of 30,000 examples.

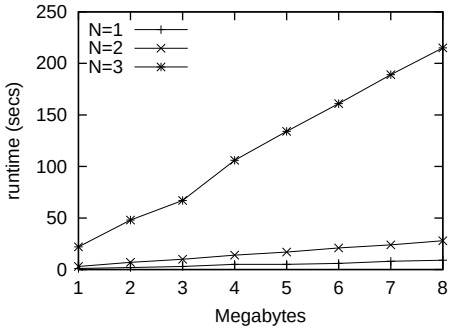


Figure 16: Increasing size of dataset and size of treatments. Datasets generated from the COCOMO model.

2. TAR2 has the potential to scale to large datasets. Assuming constant treatment size, TAR2's runtimes are linear on dataset size: see Figure 16.

3. However, the algorithm is exponential on treatment size: see the marked increase in the runtimes between N=2 and N=3 in Figure 16 and the log-linear plot of Figure 17.

The exponential impact of increasing treatment size is not necessarily a reason to reject TAR2. Firstly, if very large treatments are required, then a incremental treatment learning approach, such as used in the satellite case study of Figure 14, may suffice.

Secondly, if most domains don't need large treatments, then this exponential impact will not be seen in practice. Elsewhere, we have made an average case mathematical analysis of the ratio of the odds of a domain narrow funnels to the odds of larger funnels. Under a wide variety of assumptions, the same effect holds: the odds of narrower funnels are millions of times more likely that wider funnels [21]. Such a statistical analysis represents an average case result and may not apply to a particular domain. What would be useful would be some kind of assessment tool that checks if this average case statistical result applies to a particular domain.

The confidence1 distribution can be used to test for narrow funnels. Domains that contain such funnels would exhibit the following property: a small number of variables within the funnel exert a disproportionately large influence on the overall behavior of the system. A test for such variables is to check for *small right tails* in the confidence1 distributions. Figure 5 has such a *small right tail*; i.e. the bulk of the distribution lies away from the maximum value. Distributions with a *large right tail* such as Figure 18 are not consistent with narrow funnels. Figure 19 shows the confidence1 distributions seen in eight example sets: four from the UC Irvine repository and some of the other domains described above. Note that in all cases, the distribution has a small right tail; i.e. a small number of variables exert a disproportionately large influence on the overall behavior of the system. In all, we have applied TAR2 to 20 domains: the ones discussed in this paper and others not shown for space reasons. In none of those domains have we observed a large right tail.

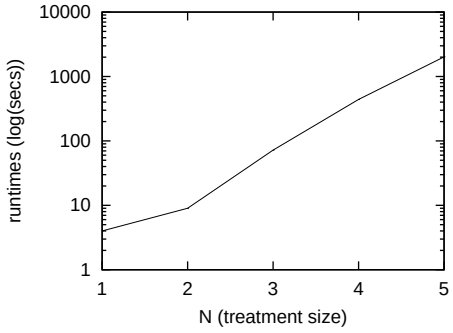


Figure 17: For different treatment sizes N, Increasing size of treatments, keeping data set size constant (3MB). Dataset generated from the COCOMO model.

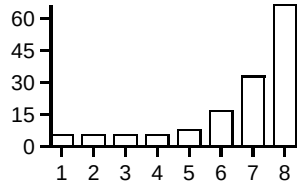


Figure 18: A hypothetical confidence1 frequency distribution with a large left tail that is inconsistent with narrow funnels. Note: yet to be observed in any example set.

7. CONCLUSION

ML-lite will not work if domains contain complex relationships. Domains with narrow funnels are not complex: the key controllers for the whole space are merely the few variables in the funnel.

The ML-lite TAR2 association rule learner is both a test and an application of funnel theory. TAR2 offers two tests for narrow funnels. Firstly, a confidence1 distribution with a small right tail is consistent with a domain containing narrow funnels. Secondly, if a domain contains narrow funnels, then TAR2 should be able to generate adequate controllers and monitors for that domain. All the domains we have seen to date have these two features.

The open issue is how many other domains lack complex relationships. Based on around 20 case studies with TAR2 (some of which were reported above), and Holte's prior work with 1R, we have some empirical reasons to believe that many domains are not complex. Also, we have theoretical reasons for believing that nar-

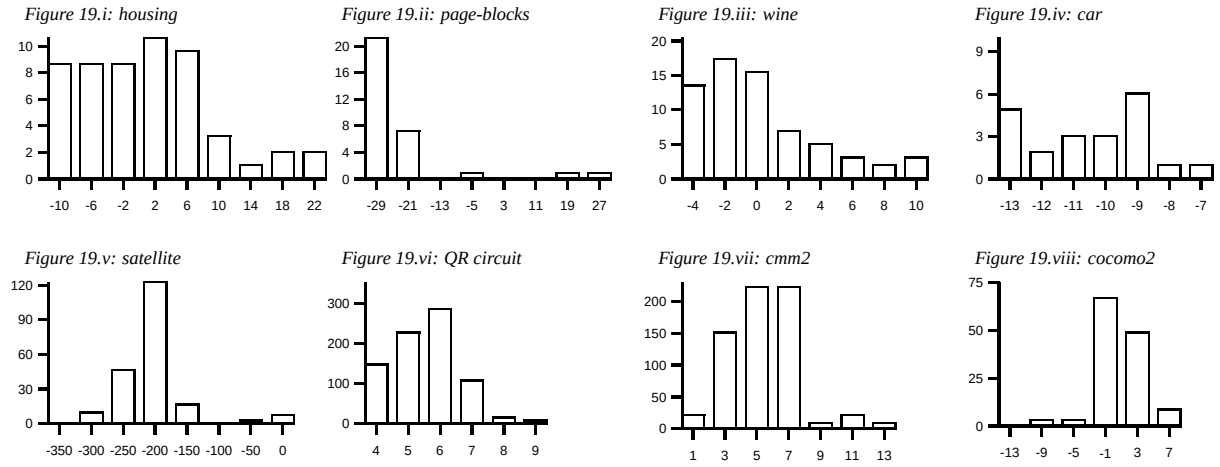


Figure 19: Confidence1 distributions seen in eight domains. Y-axis is the number of times a particular confidence1 was seen. Top row comes from datasets taken from the UC Irvine repository. Bottom row were generated from other domains discussed in this article.

row funnels are common [16,17,21,23]; i.e. domains with complex relationships are rare and ML-lite methods like TAR2 will often suffice.

The success of such a simple algorithm such as TAR2 suggests that it can be fruitful to *first* try lightweight methods *before* exploring heavyweight methods. We advocate using TAR2 as a preprocessor to other, more elaborate schemes. If TAR2 can learn adequate controllers and monitors, then the domain is simple enough for ML-lite. Otherwise, ML-heavy methods should be explored.

TAR2 is available for download from `http://www.ece.ubc.ca/twiki/bin/view/Softeng/TreatmentLearner`.

8. REFERENCES

[1] C. Abts, B. Clark, S. Devnani-Chulani, E. Horowitz, R. Madachy, D. Reifer, R. Selby, and B. Steece. COCOMO II model definition manual. Technical report, Center for Software Engineering, USC,, 1998. `http://sunset.usc.edu/COCOMOII/cocomox.html#downloads`.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Databases*, 1994. Available from `http://www.almaden.ibm.com/cs/people/ragrawal/papers/vldb94_rj.ps`.

[3] S.B. Bay and M.J. Pazzani. Detecting change in categorical data: Mining contrast sets. In *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*, 1999. Available from `http://www.ics.uci.edu/~pazzani/Publications/stucco.pdf`.

[4] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. Classification and regression trees. Technical report, Wadsworth International, Monterey, CA, 1984.

[5] C.H. Cai, A.W.C. Fu, C.H. Cheng, and W.W. Kwong. Mining association rules with weighted items. In *Proceedings of International Database Engineering and Applications Symposium (IDEAS 98)*, August 1998. Available from `http://www.cse.cuhk.edu.hk/~kdd/assoc_rule/paper.pdf`.

[6] D.J. Clancy and B.K. Kuipers. Model decomposition and simulation: A component based qualitative simulation algorithm. In *AAAI-97*, 1997.

[7] J. Crawford and A. Baker. Experimental results on the application of satisfiability algorithms to scheduling problems. In *AAAI '94*, 1994.

[8] J. DeKleer. An Assumption-Based TMS. *Artificial Intelligence*, 28:163–196, 1986.

[9] M.S. Feather, S.L. Cornford, and T.W. Larson. Combining the best attributes of qualitative and quantitative risk management tool support. In *15th IEEE International Conference on Automated Software Engineering, Grenoble, France*, pages 309–312, September 2000.

[10] M.S. Feather and T. Menzies. Converging on the optimal attainment of requirements. In *RE'03 (submitted)*, 2002.

[11] R.C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11:63, 1993.

[12] J.R. Josephson, B. Chandrasekaran, M. Carroll, N. Iyer, B. Wasacz, and G. Rizzoni. Exploration of large design spaces: an architecture and preliminary results. In *AAAI '98*, 1998. Available from `http://www.cis.ohio-state.edu/~jj/Explore.ps`.

[13] B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *KDD*, pages 80–86, Sept 1998. Available from `http://citeseer.nj.nec.com/liu98integrating.html`.

[14] R. Lutz and R. Woodhouse. Bi-directional analysis for certification of safety-critical software. In *1st International Software Assurance Certification Conference (ISACC'99)*, 1999. Available from `http://www.cs.iastate.edu/~rlutz/publications/isacc99.ps`.

[15] R. Madachy. Heuristic risk assessment using cost factors. *IEEE Software*, 14(3):51–59, May 1997.

[16] T. Menzies and B. Cukic. Adequacy of limited testing for knowledge based systems. *International Journal on Artificial Intelligence Tools (IJAIT)*, June 2000. Available from `http://tim.menzies.com/pdf/00ijait.pdf`.

[17] T. Menzies and B. Cukic. When to test less. *IEEE Software*, 17(5):107–112, 2000. Available from `http://tim.menzies.com/pdf/00iesoft.pdf`.

[18] T. Menzies and Y. Hu. Constraining discussions in requirements engineering. In *First International Workshop on Model-based Requirements Engineering*, 2001. Available from `http:`

//tim.menzies.com/pdf/01lesstalk.pdf.

[19] T. Menzies and Y. Hu. Reusing models for requirements engineering. In *First International Workshop on Model-based Requirements Engineering*, 2001. Available from http://tim.menzies.com/pdf/01reusere.pdf.

[20] T. Menzies and Y. Hu. Agents in a wild world. In C. Rouff, editor, *Formal Approaches to Agent-Based Systems, book chapter*, 2002. Available from http://tim.menzies.com/pdf/01agents.pdf.

[21] T. Menzies and H. Singh. Many maybes mean (mostly) the same thing. In *2nd International Workshop on Soft Computing applied to Software Engineering (Netherlands), February*, 2001. Available from http://tim.menzies.com/pdf/00maybe.pdf.

[22] T. Menzies and E. Sinsel. Practical large scale what-if queries: Case studies with software risk assessment. In *Proceedings ASE 2000*, 2000. Available from http://tim.menzies.com/pdf/00ase.pdf.

[23] Tim Menzies, Bojan Cukic, Harhsinder Singh, and John Powell. Testing nondeterminate systems. In *ISSRE 2000*, 2000. Available from http://tim.menzies.com/pdf/00issre.pdf.

[24] T.J. Menzies and P. Compton. Applications of abduction: Hypothesis testing of neuroendocrinological qualitative compartmental models. *Artificial Intelligence in Medicine*, 10:145–175, 1997. Available from http://tim.menzies.com/pdf/96aim.pdf.

[25] T.J. Menzies, S. Easterbrook, Bashar Nuseibeh, and Sam Waugh. An empirical investigation of multiple viewpoint reasoning in requirements engineering. In *RE '99*, 1999. Available from http://tim.menzies.com/pdf/99re.pdf.

[26] A. Parkes. Lifted search engines for satisfiability, 1999.

[27] R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufman, 1992. ISBN: 1558602380.

[28] R. Rymon. An se-tree-based prime implicant generation algorithm. In *Annals of Math. and A.I., special issue on Model-Based Diagnosis*, volume 11, 1994. Available from http://citeseer.nj.nec.com/193704.html.

[29] Ron Rymon. An SE-tree based characterization of the induction problem. In *International Conference on Machine Learning*, pages 268–275, 1993.

[30] Josh Singer, Ian P. Gent, and Alan Smaill. Backbone fragility and the local search cost peak. *Journal of Artificial Intelligence Research*, 12:235–270, 2000.

[31] K. Wang, Y. He, D. Cheung, and F. Chin. Mining confident rules without support requirement. In *10th ACM International Conference on Informationand Knowledge Management (CIKM 2001), Atlanta*, 2001. Available from http://www.cs.sfu.ca/~wangk/publications.html.

[32] G. Webb. Efficient search for association rules. In *Proceeding of KDD-2000 Boston, MA*, 2000. Available from http://citeseer.nj.nec.com/webb00efficient.html.