Technische Universität Darmstadt
Knowledge Engineering Group
Hochschulstrasse 10, D-64289 Darmstadt, Germany

http://www.ke.informatik.tu-darmstadt.de

**Technical Report TUD–KE–2008–02**

*Frederik Janssen, Johannes Fürnkranz*

# An Empirical Comparison of Hill-Climbing and Exhaustive Search in Inductive Rule Learning

# An Empirical Comparison of Hill-Climbing and Exhaustive Search in Inductive Rule Learning

**Frederik Janssen** JANSSEN@KE.INFORMATIK.TU-DARMSTADT.DE
**Johannes Fürnkranz** FUERNKRANZ@INFORMATIK.TU-DARMSTADT.DE
*Knowledge Engineering Group*
*Department of Computer Science*
*TU Darmstadt, Germany*

## Abstract

Most commonly used inductive rule learning algorithms employ a hill-climbing search, whereas local pattern discovery algorithms employ exhaustive search. In this paper, we evaluate the spectrum of different search strategies to see whether separate-and-conquer rule learning algorithms are able to gain performance in terms of predictive accuracy or theory size by using more powerful search strategies like beam search or exhaustive search. Unlike previous results that demonstrated that rule learning algorithm suffer from over-searching, our work pays particular attention to the connection between the search heuristic and the search strategy, and we show that for some rule evaluation functions, complex search algorithms will consistently improve results without suffering from the over-searching phenomenon. In particular, we will see that this is typically the case for heuristics which perform bad in a hill-climbing search. We interpret this as evidence that commonly used rule learning heuristics mix two different aspects: a rule evaluation metric that measures the predictive quality of a rule, and a search heuristic that captures the potential of a candidate rule to be refined into highly predictive rule. For effective exhaustive search, these two aspects need to be clearly separated.

## 1. Introduction

Most classification rule learning algorithms use hill-climbing as their method for greedily adding conditions to a rule, whereas local pattern discovery algorithms, such as association rule or subgroup discovery often use some form of exhaustive search. Beam search can be viewed as a means for trading off between these two. The question that we pose in this paper is how the quality of the found theories changes with increased search effort.

Several authors have previously observed a phenomenon called *over-searching*, which essentially says that increased search effort will not only not improve the results but may even lead to a decrease in accuracy. For example, Murthy and Salzberg (Murthy & Salzberg, 1995) have found that increasing the look-ahead in decision tree induction will typically no longer improve the results, and may also produce larger and less accurate trees. Specifically for inductive rule learning, Quinlan and Cameron-Jones (Quinlan & Cameron-Jones, 1995) showed that more search has not to lead to better predictive accuracy. However, their work was limited to the use of a single heuristic for evaluating rules, the *Laplace* error.

Following this direction, our work aims at re-evaluating the over-searching problem for many other heuristics and on different datasets. The key difference to the previous work is that we evaluate nine different heuristics which have been recently evaluated for heuristic

search (Janssen & Fürnkranz, 2008). Most of them are well-known for heuristic search, but have never been used for exhaustive search before. Collectively, they span a wide variety of different biases for evaluating a single rule. We will show that the search mechanism is interweaved directly with the search heuristic. Our results confirm the previous results, but we argue that the over-searching phenomenon depends on the used heuristic. This paper gives evidence that, for several search heuristics, a complete search does not only result in a smaller theory, as also observed in (Quinlan & Cameron-Jones, 1995), but also in a more accurate one. This is particularly likely to happen for heuristics that perform badly in hill-climbing search, while other heuristics, that perform rather well, will lead to a decrease in performance when used in exhaustive search. The main conclusion that we will draw from this investigation is that heuristics that are tailored to hill-climbing search, also have to capture the potential that the current rule can be refined in to a rule with a high predictive quality, and that it is mainly this aspect that is responsible for these observed performance differences.

We begin the paper with a brief recapitulation of *separate-and-conquer* rule learning, and, in particular, describe our implementation in some detail. In Section 3, we discuss the three different search strategies. Thereafter, the rule learning heuristics we used for the study are summarized in Section 4. The experimental setup is described in Section 5 and the results of the experiments are presented in Section 6. Section 7 gives a conclusion of the work conducted in this study.

**Algorithm 1** FindBestRule*(Examples,h)*

```
InitRule = ∅
InitVal = EvaluateRule(InitRule,Examples,h)
BestRule = <InitVal,InitRule>
Rules = {BestRule}
while Rules ≠ ∅
    Candidates = SelectCandidates(Rules, Examples)
    Rules = Rules \ Candidates
    for Candidate ∈ Candidates
        Refinements = RefineRule(Candidate, Examples)
        for Refinement ∈ Refinements
            Evaluation = EvaluateRule(Refinement,Examples,h)
            NewRule = <Evaluation,Refinement>
            Rules = InsertSort(NewRule, Rules)
            if NewRule > BestRule
                BestRule = NewRule
    Rules = FilterRules(Rules, Examples)
return(BestRule)
```

## 2. Separate-and-Conquer Rule Learning

The goal of an inductive rule learning algorithm is to automatically learn rules that allow to map the examples of a domain to their respective classes. Algorithms differ in the way they learn individual rules, but most of them employ a *separate-and-conquer* or *covering* strategy for combining rules into a rule set (Fürnkranz, 1999). For the sake of the reproducibility of the results, we use this section to describe our particular implementation. It can be safely skipped by readers that are primarily interested in the experimental results.

### 2.1 Overview of the strategy

Separate-and-conquer rule learning can be divided into two main steps: First, a single rule is learned from the data (the *conquer* step) by a procedure called FindBestRule. Following (Fürnkranz, 1999), Figure 1 shows a generic version of this procedure which can be instantiated into various specific search algorithms. In particular, it can simulate the three strategies that we will describe in Section 3.

The procedure FindBestRule searches the hypothesis space for a rule that optimizes a given quality criterion $h$. It maintains *Rules*, a sorted list of candidate rules, which is initialized with an empty set of conditions. New rules will be inserted in appropriate places (InsertSort), so that *Rules* will always be sorted in decreasing order of the heuristic evaluations $h$, which are determined by EvaluateRule. At each cycle, SelectCandidates selects a subset of these candidate rules, which are then refined using RefineRule. In our case, a refinement is the addition of an attribute-value test. Each refinement is evaluated and inserted into the sorted *Rules* list. If the evaluation of the *NewRule* is better than the best rule found previously, *BestRule* is set to *NewRule*. FilterRules selects the subset of the ordered rule list that will be used in subsequent iterations, and, when all candidate rules have been processed, returns the best encountered rule. The three search strategies used in this paper, hill-climbing, beam search, exhaustive search, can be realized by allow-

ing FilterRules to let only the best $n$ refinements pass for the next iteration ($n$ is the beam width, $n = 1$ results in beam search, and $n = \infty$ produces an (inefficient) exhaustive search). For hill-climbing and beam search, SelectCandidates will always return all *Rules*, whereas an exhaustive search will only look at the first element in this sorted list.

Then, the main covering loop removes all examples that are covered by the learned rule from the training set (the *separate* step), and the next rule is learned on the remaining examples. The two steps are repeated as long as positive examples are left in the training set. This ensures that every positive example is covered at least by one rule (*completeness*) and no negative example is included (*consistency*). The origin of this strategy is the AQ-Algorithm (Michalski, 1969) but it is still used in many algorithms, most notably in Ripper (Cohen, 1995), arguably the most accurate rule learning algorithms today.

### 2.2 Implementation

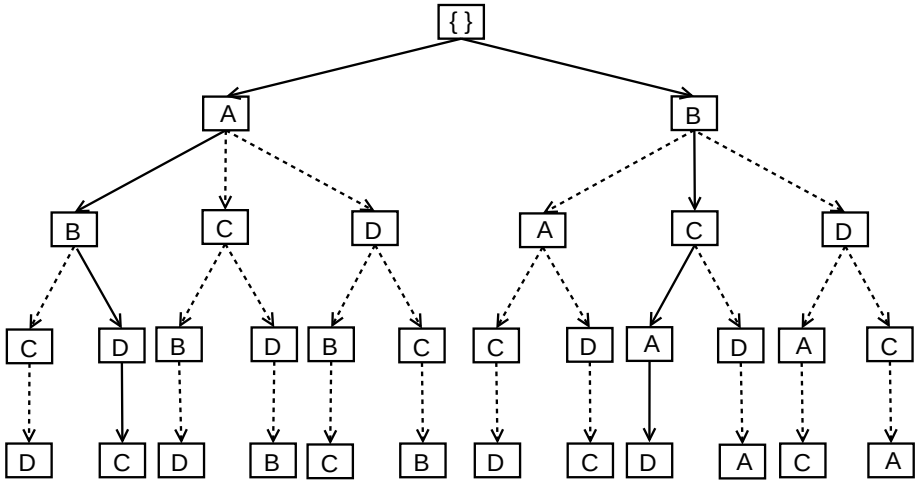
For our experiments, we implemented a simple separate-and-conquer algorithm with a top-down strategy for selecting individual rules. The rule refinement process follows closely the general procedure described in Algorithm 1. At the moment, the algorithm can only handle nominal attributes, we are currently working on an implementation that can also use numeric attributes.

The algorithm can be ran with different heuristics and is able to handle multi-class problems by employing a simple *one-against-all* class binarization where the classes are ordered in ascending frequency. Thus all instances that belong to the current class are treated as positive examples and all other examples represent the negative class label. When the algorithm has finished learning $c - 1$ of $c$ classes it predicts the most frequent class (the last one) as default class. For classification of a new example, each rule in the (decision) list is checked whether it covers the example or not and the first one that "fires" is used to classify the example. If no rule in the list covers the example the default class is predicted. The heuristic value of this exceptional rule is also used as a minimum quality measure for all rules. If no rule could be found that outperforms the default rule the theory remains empty. Missing values are treated as examples that are never covered. For rules with equal evaluation a tie breaking on the covered positive examples is done. This is especially important for the heuristic *Precision* and is discussed in detail in Section 6.5.

Our algorithm does not feature a special pruning or optimization phase. Nevertheless some pruning is done in both loops: the outer covering loop does not add the rule *BestRule* to the theory if the rule returned by FindBestRule is empty or if it covers fewer positive than negative examples. When the negative coverage is higher than the positive a rule will never increase the accuracy of the theory. The procedure FindBestRule stops refining the current rule *NewRule* if no negative examples are covered or a simple forward pruning criterion fires.

*Forward pruning* (also called pruning with "optimistic value" in (Webb, 1995)) is used to cut off subtrees of the search space without losing performance. In the used implementation it works as follows: Assume the current rule *NewRule* covers $p$ positive and $n$ negative examples. The best rule that could be yielded by this refinement process would be one that covers $p$ positives (does not loose a positive) and 0 negatives (excludes all negatives). If $h(NewRule) \leq h(BestRule)$ *NewRule* is not refined any more (not inserted into *Rules*).

Figure 1: Hill-Climbing and Beam Search




## 3. Search Strategies

Most global rule learning algorithms employ a hill-climbing strategy (Lavrač, Flach, Kasek, & Todorovski, 2002; Cohen, 1995) whereas algorithms for association rule discovery typically perform an exhaustive search that discovers all patterns that satisfy a given set of constraints. The first technique starts with a rule that covers all examples (the empty rule) and evaluates all possible conditions. The best one according to a heuristic measure is selected, all others are discarded and based on this first condition all candidate rules with two conditions are generated. The second mechanism at each conquer step generates and evaluates all rules that can be built from the data and selects the best one.

To trade off between these two methods usually a beam search is used where a parameter $n$ determines how many rules are refined in a single steps.

Another method for searching good theories is to generate all possible theories rather than search all single rules. The computational demands of this approach are far beyond the mechanism that only generates all rules. Therefore we concentrated on evaluating different strategies for searching a single rule and left the main covering loop of the algorithm untouched.

### 3.1 Hill-Climbing

Hill-climbing represents the instantiation of a beam search with beam size $n = 1$. The advantage of this method clearly lies in its efficiency concerning both memory and time issues. The disadvantage is that the search can be stuck in a local optimum without finding the global optimum. Note that for any refinement process there is exactly one path through the search space. In the first step of the FindBestRule procedure all attribute-value pairs are generated and evaluated. Until then the used conditions are stored so that they are not generated twice.

Figure 1 (left) shows an example for a refinement path through the search space. Assume that there are only the attributes A and B that can be selected in the first test (attribute values are omitted). Hill-climbing now adds the attribute with the highest evaluation value (in this case A). In the second step there are only 3 possible Tests (B,C and D). Attribute D is selected, B and C are remaining, hill-climbing selects B and then only C remains. Note that now the refinement process is terminated because no more attributes exist. The returned rule has not to be $A \wedge D \wedge B \wedge C \rightarrow CLASS$ since any of the previously generates ones can have a higher evaluation value and thus would be returned.

### 3.2 Beam Search

Beam search is useful for avoiding situations where the locally optimal choices of hill-climbing makes a globally suboptimal choice. The idea is simply to refine $n$ rules simultaneously. If $n \rightarrow \infty$ the beam search turns into an exhaustive search. There is some work on determining a good beam size for a single dataset (Quinlan & Cameron-Jones, 1995) but it is still an open question how to choose $n$. In our experiments we simply tested the beam sizes $1, 2, 4, ..., 2048$.

Our implementation of the beam search generates all attribute-value pairs in each step but only adds a rule if it is not contained in the current beam. Thus, the search space is basically unordered due to many possible refinement paths starting from different attribute tests. Therefore, the algorithms only remembers what attributes are already used for one single rule because for nominal attributes a test of an attribute should only occur once in a rule.

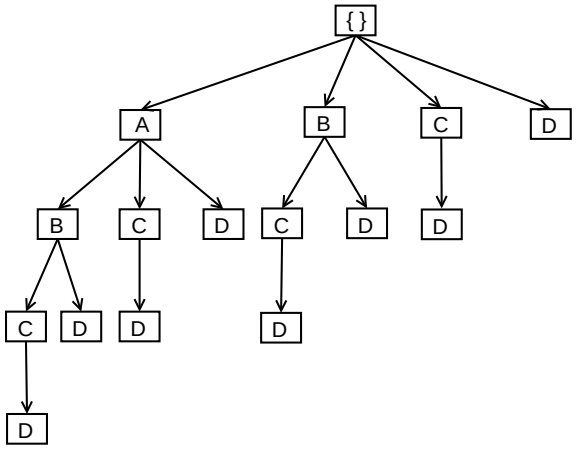
Figure 1 also displays an example for a beam search with a size of 2. Assume the same start situation as in the example before. Beam Search is able to both refine A and B. It adds conditions in the same way as done in hill-climbing. At last both rules contain 4 conditions and are semantically the same rule but with a different ordering of conditions. In this case, as described above, the second one would not be added to the current beam. Instead, the rule with the second best evaluation is included in the beam.

### 3.3 Exhaustive Search

There are some global rule learning systems that incorporate an exhaustive search, most notably *OPUS* (Webb, 1995). The naive implementation sketched in Section 2.1 suffers from the problem that the same rule can be reached over multiple refinement paths, as also shown in Figure 1. Presumably for this reason, Quinlan and Cameron-Jones did not include a true exhaustive search into their comparison, but used a maximum beam width of 512.

We implemented a more efficient version, based on the ordered search algorithm of the $OPUS^o$ rule learner. As shown in Figure 2, the algorithm only generates each rule once, and therefore does not have to check if it is already contained in the beam. This is because the procedure has to construct every single rule anyway. Thus, the computational demands of the exhaustive search are much smaller than those of a beam search with $n \rightarrow \infty$ because the latter may generate multiple paths that end in the same refinement. Note that the sequence in which the rules are generated may differ from the sequence generated by the beam search. Therefore, for some datasets, the performance of the two search strategies

Figure 2: Exhaustive search




may be different, because we cannot guarantee that, among rules with identical evaluations, the same rule is selected in both algorithms.

## 4. Rule Learning Heuristics

A rule learning heuristic is a function $h$ that evaluates candidate rules for their potential of being refinable into good rules. Most commonly used heuristics base the evaluation of a rule $r$ on its coverage statistics on the training set. A good heuristic should, on the one hand, maximize the number of positive examples $p$ that are covered by the rule (coverage), and, on the other hand, the rule should cover as few negative examples $n$ as possible (consistency). A simple way of achieving both objectives is to subtract the number of covered negatives from the covered positives. The resulting heuristic ($h_{Accuracy} = p - n$) is equivalent to accuracy, which computes the percentage of correctly classified examples among all training examples ($P$ positive and $N$ negative). Other heuristics employ more complex ways to reach these two objectives.

In this study we experimented with 9 heuristics displayed in Table 1. All of these heuristics employ different strategies for evaluating a single rule. *Precision*, for example, tends to learn many rules which usually contain of a lot of conditions. On the contrary *WRA* (Lavrač, Flach, & Zupan, 1999) often settles for very few rules that are overly general (Todorovski, Flach, & Lavrač, 2000; Janssen & Fürnkranz, 2008). In this sense, each heuristic has its own strategy for navigating the search process in the right direction. In a recent study (Janssen & Fürnkranz, 2008), we have evaluated the performance of these heuristics in a hill-climbing search (with the exception of the *Odds Ratio*). In this study, we also determined parameters for the two heuristics which allow to trade off between optimizing consistency and coverage. For the $m$-estimate we have chosen $m = 22.466$, and for the *relative cost measure* we selected $c = 0.342$ as recommended in (Janssen & Fürnkranz, 2008) for hill-climbing search. The final heuristic has been *Meta-learned* using linear regression on various characteristics of the training set, such as the positive and negative coverage and the precision of the rule, the class distribution of the problem, etc. It tries to predict the

Table 1: The used heuristics

| heuristic | formula |
|---|---|
| *Precision* | $\frac{p}{p+n}$ |
| *Laplace* | $\frac{p+1}{p+n+2}$ |
| *m-estimate* | $\frac{p+m\cdot\frac{P}{P+N}}{p+n+m}$ |
| *Weighted Relative Accuracy* | $\frac{p}{P}-\frac{n}{N}$ |
| *Accuracy* | $p-n$ |
| *Relative Cost Measure* | $c\cdot\frac{p}{P}-(1-c)\cdot\frac{n}{N}$ |
| *Odds ratio* | $\frac{p\cdot(N-n)}{(P-p)\cdot n}$ |
| *Correlation* | $\frac{p\cdot(N-n)-n\cdot(P-p)}{\sqrt{P\cdot N\cdot(p+n)\cdot(P-p+N-n)}}$ |
| *Meta-learned* | meta-learned combination of $p$, $n$, $P$ and $N$ |

out-of-sample precision of the final rule, based on training set characteristics of the current rule. For a detailed description we refer to (Janssen & Fürnkranz, 2007, 2008).

It should be noted that classical rule evaluation metrics typically focus on evaluating the discriminatory power of a rule. However, if we consider the learning of a rule as a search problem, as we do in this work, a rule should rather be evaluated by its potential of being refined into a such a rule. In particular for the parametrized and meta-learned heuristics mentioned above, the parameter has been optimized in the context of a hill-climbing algorithm, and will implicitly take this into account.

## 5. Experimental Setup

Our goal was to experiment on a large number of data sets with many different beam sizes. The primary interest was how the accuracy achieved in average on all datasets varies. To illustrate the effects of increasing beam sizes we also experimented with a rule learner that only learns one single rule for each class. Ideally, these effects are illustrated best on datasets where local minima occur or which are hard to learn. As there are some datasets that show strong performance variations among different beam sizes, we also include some plots of individual datasets.

Some of the commonly used UCI datasets (Newman, Blake, Hettich, & Merz, 1998) were too big to use (in terms of attribute-value pairs and classes) due to the vast memory demands of larger beam sizes. The datasets we used were *autos-d, balloons, breast-cancer, breast-w-d, bridges2-d, colic.ORIG-d, contact-lenses, hayes-roth, hepatitis-d, monk1, monk2, monk3, mushroom, primary-tumor, promoters, solar-flare, soybean, tic-tac-toe, titanic, vote-1, vote, zoo*. We focused on datasets with primarily nominal attributes, but also included some that contained numeric attributes (marked with a "-d"). In this case, the numeric attributes were discretized into 10 different values, using equal-width discretization.

On each dataset a *10-fold stratified Cross Validation* implemented in *Weka* (Witten & Frank, 2005) was used to obtain performance statistics for all different combinations. As a crude measure for comparing the performance of heuristics we use *macro-averaged accuracy*

(the average fraction of correctly classified examples over all datasets). However, we will also look at the behavior of individual datasets. We do not report training set accuracy, as it will typically increase with increased search effort.

We also report the *size* of the theory, typically in terms of numbers of conditions of all rules. Especially in Section 6.1 the number of rules is also used. Along with these we employed a *ranking* based on the accuracy of the heuristics. Note that the ranking handles heuristics that achieve the same accuracy by dividing the number of them by the sum of their "standard" positions, i.e., for three heuristics that have the same accuracy their rank would be $\frac{3}{1+2+3} = 0.5$. The ranking gives additional information about the quality of the heuristic that sometimes are not detectable when only using accuracy values. Often there are large differences in the variances of the accuracies of individual datasets that are neglected when the ranking is used as evaluation method. Finally, we also report *runtime* measurements.

## 6. Results

This section provides a detailed discussion of our empirical study. We examine each single heuristic in terms of predictive accuracy, theory size (number of conditions) and the runtime of each metric (Section 6.1). Subsequently, results on single data sets are discussed (Section 6.3). To illustrate how the rules change when they are searched more exhaustively we also included a version of the algorithm where the main covering loop is only iterated once, so that there is one single rule for each class (Section 6.4). Following from the previous observations we identify several interesting properties of Covering algorithms used with complex search mechanisms which are discussed in Section 6.5.

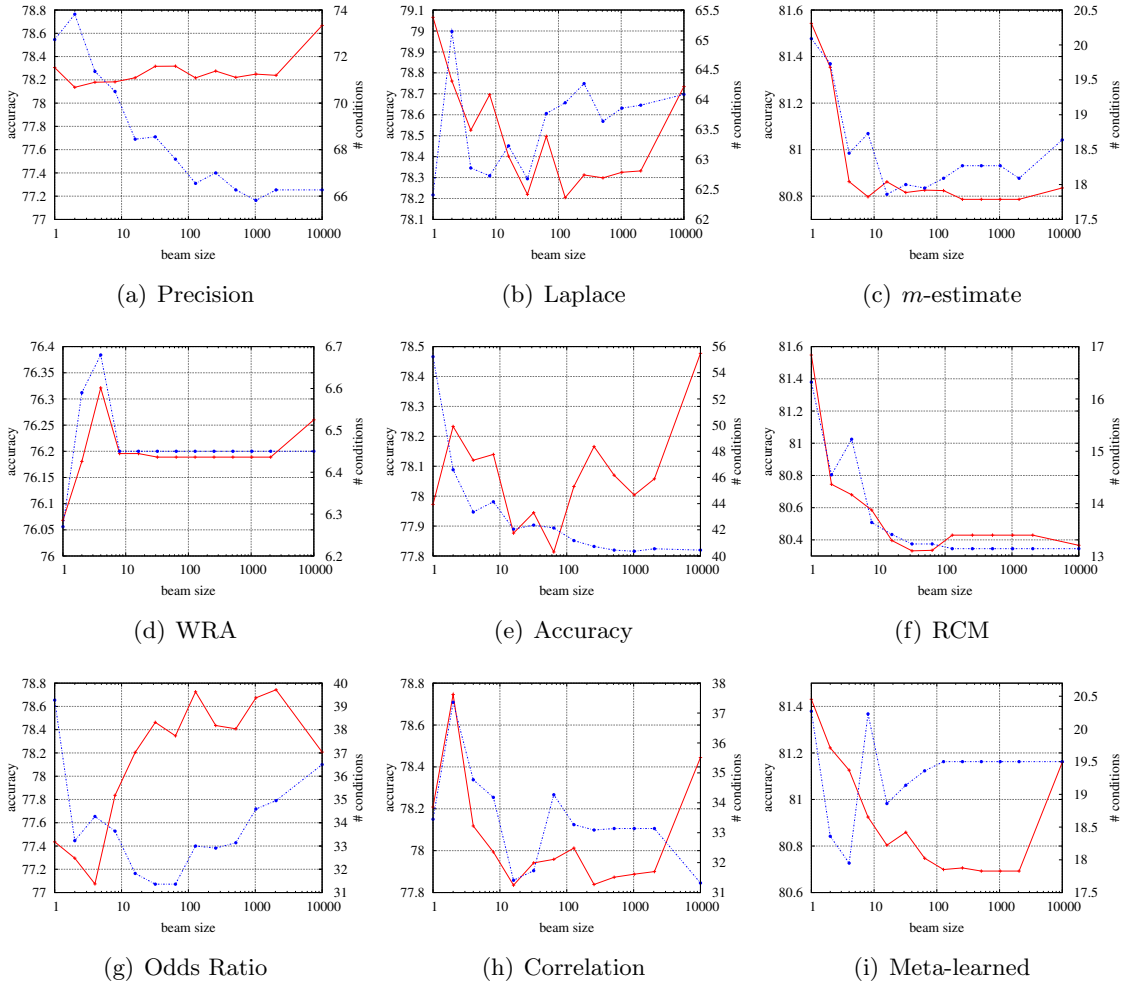
### 6.1 Varying the Beam Size

Figure 3 displays the results in terms of accuracy (left y-axis) and number of conditions (right y-axis) for all heuristics. The x-axis displays the varied beam sizes in logarithmic scaling. A beam size of 10000 is used for denoting exhaustive search. The solid (red) line displays the average accuracy and the dotted (blue) line the number of conditions over various beam sizes.

The results for the heuristics are quite different. Some of them show a clear profit of the simultaneous refinement of more than one search path. *Precision*, for example, is able to constantly gain performance (with some fluctuation) and simultaneously decreases the theory size. This is a bit surprising, as one might expect that exhaustive search will be more likely to discover overfitting rules that cover only a few positive and no negative examples. However, the opposite seems to be the case: exhaustive search is able to discover pure rules that are shorter and have a higher coverage (it should be noted that positive coverage is used as a tie breaker when multiple rules have the same evaluation).

*Laplace*, on the other hand, shows much more fluctuation and ends up with a lower accuracy than those reached with simple hill-climbing. This, essentially, confirms the results of (Quinlan & Cameron-Jones, 1995). Interestingly, contrary to the results of *Precision*, the theory size seems to increase steadily (with the exception of the outlier for a beam size of 2). However, both *Precision* and *Laplace* seem to arrive at very similar theories with about 65 conditions on average, and an average predictive performance of about 78.7%. Apparently,

Figure 3: Accuracies vs. Number of Conditions for all heuristics




(a) Precision (b) Laplace (c) $m$-estimate

(d) WRA (e) Accuracy (f) RCM

(g) Odds Ratio (h) Correlation (i) Meta-learned

while the *Laplace* measure is effective in preventing overfitting for hill-climbing algorithms, its performance degrades to the performance of *Precision*.

*Weighted Relative Accuracy (WRA)*, on the other hand, is a very stable heuristic. The reason for this behavior is that it effectively over-generalizes. It learns by far the smallest theories (only about 6.5 conditions on average), with a performance that lags considerably behind the performance of the above two heuristics. Thus, the optimal rules are typically found at very shallow search depths and comparably small beam sizes.

The final minor jump in predictive accuracy when moving from a beam size of 2048 to exhaustive search is due to the different order of the exploration of the search space with beam search and exhaustive search (cf. Figure 1 and 2). As a result, the two algorithms may pick different rules when they have the same heuristic value and the same tie-breaking statistics, which in turn, due to the covering loop, may result in entirely different theories. Apparently, this happens several times in the cross-validations of the 22 datasets, producing this minor deviation. This effect also has to be taken into account in several other cases, but

Table 2: Runtimes (in sec.) of the heuristics with different beam sizes

| beam size | Precision | Laplace | $m$-estimate | WRA | Accuracy | RCM | Odds Ratio | Correlation | Meta-learned |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 884.2 | 808.4 | 495.6 | 299.2 | 741.3 | 234.5 | 615.0 | 586.3 | 457.5 |
| 2 | 1405.1 | 1243.7 | 632.3 | 362.7 | 1024.6 | 334.7 | 793.2 | 836.8 | 627.9 |
| 4 | 1447.4 | 1292.6 | 684.7 | 386.0 | 1037.3 | 360.6 | 832.3 | 938.2 | 710.6 |
| 8 | 1520.8 | 1423.8 | 729.4 | 431.3 | 1063.1 | 359.0 | 890.9 | 1007.5 | 764.8 |
| 16 | 1586.9 | 1571.1 | 817.0 | 439.9 | 1141.1 | 363.0 | 920.0 | 1074.7 | 826.8 |
| 32 | 1721.7 | 1787.1 | 879.1 | 481.3 | 1211.2 | 376.8 | 981.6 | 1170.4 | 924.1 |
| 64 | 1891.5 | 1950.7 | 935.6 | 503.1 | 1288.1 | 394.2 | 1112.6 | 1291.8 | 1074.0 |
| 128 | 2102.4 | 2098.5 | 1066.1 | 577.2 | 1443.1 | 412.0 | 1387.6 | 1495.1 | 1218.1 |
| 256 | 2466.0 | 2518.8 | 1279.7 | 679.0 | 1660.5 | 436.2 | 1651.1 | 1861.2 | 1445.1 |
| 512 | 3380.5 | 3341.6 | 1670.2 | 786.9 | 2061.3 | 480.9 | 2042.8 | 2379.7 | 1871.4 |
| 1024 | 4626.4 | 5183.8 | 2344.2 | 976.1 | 2727.1 | 526.0 | 2802.2 | 3300.6 | 2885.0 |
| 2048 | 8227.3 | 7638.2 | 3395.8 | 1473.0 | 3904.4 | 576.3 | 4076.6 | 5174.1 | 3903.2 |
| exh. | 12061.1 | 11764.1 | 3078.9 | 928.8 | 5596.6 | 1714.3 | 4310.4 | 5076.4 | 4359.7 |

it should also be noted that in many cases (e.g., *Accuracy*), we can still observe changes at beam sizes $> 1000$, so that the increase in performance when moving from beam sizes 2048 to exhaustive search, which can be observed in 7 of the 9 heuristics, has some credibility.

In terms of accuracy gain the *Odds Ratio* works best with 1.3% average accuracy gain from hill-climbing to the best working beam size of 2048. The theory size could be decreased and has its minimum at a beam size of 64 where the corresponding accuracy has already exceeded those of hill-climbing. The *Accuracy* heuristic also shows a big performance gain of 0.51% average accuracy that goes along with a decrease of conditions of 14.78.

Interestingly all three heuristics that were optimized for hill-climbing (*m-estimate, relative cost measure (RCM)*, and the *Meta-learned* heuristic) do not perform well under deeper searches. In all cases, exhaustive search finds simpler theories than greedy search, but these are of a lesser quality. This is consistent with several previous results that show that contrary to the assumptions of *Occam's razor*, simpler theories often exhibit a worse performance (we refer to (Domingos, 1999) for a summary of such results). We explain these results with the fact that these heuristics have been optimized for hill-climbing, and that they thus implicitly take the search process into account. As discussed above, a good heuristic for a hill-climbing search should try to predict the quality of the best rule to which it can be refined to, in order to make sure that the path to the best final rule can be found. This, on the other hand, is not necessary for exhaustive search, where we are guaranteed to find the best rule.

### 6.2 Runtime of the search methods

Table 2 shows the runtimes of the different methods in seconds. Not surprisingly, the runtime generally increases for more exhaustive searches. However, it is interesting to look at the relative increase of time for consecutive beam sizes. Some measurements are practically the same (cf. *Accuracy* with beam size 2 to 16) which means that the number of evaluated rules does not change much. This, again, reflects the fact that some heuristics prefer general rules, and these are already found with low beam sizes.

On the other hand, both *Precision* and *Laplace* have a strong bias towards overfitting rules which is also reflected in their runtimes. As the beam grows, the runtime of the algorithm increases. When changing the search to an exhaustive one the time grows again. *Precision* with over 3 hours takes the maximum amount of total runtime among all heuristics.

Note that the implementation of the exhaustive search sometimes even is more efficient than the beam search with $n = 2048$. But this is mostly with heuristics that do not induce many candidate rules as described above.

### 6.3 Results for Individual Datasets

Of course, macro-averaged accuracy over several datasets is a very crude and not very meaningful summary measure that we only used because of the lack of a better alternative. To illustrate the effects of the different search mechanisms without averaging the values, Figure 4 displays results of all interesting heuristics (those where some changes happen) on six selected datasets. The $x$-axis displays the beam size in a logarithmic scaling (beam size of 10000 encodes exhaustive search), and the $y$-axis depicts the cross-validated accuracy on this dataset. Most of the heuristics show some fluctuation between different different beam sizes, but typically a clear trend can be recognized.

There are some heuristics, primarily *WRA*, that remain quite constant over all sizes and data sets, for reasons we already discussed above. Thus, we only included this heuristic in the plot for "monk2" because its performance there was superior. After a degradation of 1.78% in accuracy it achieves the highest value with a beam size of 8 and from then on does not change any more. The strongest variations are visible on the data set "breast-cancer" where only *Correlation* and the *Relative Cost Measure* remain fairly constant.

The set "autos-d" is one example for the constant performance gain of the heuristic *Odds Ratio*, but also of its bad performance in hill-climbing. Until a beam size of 64 it profits from bigger beam sizes before the accuracy falls down again. High beam sizes $> 512$ show a comparable accuracy but the theory size dropped by 2 conditions. The data set also is a good example for the different behavior of the heuristics. As mentioned above *Odds Ratio* constantly gains performance, has an optimum at a beam size of 64 (22.93% accuracy gain) and ends up about 17% more accurate than with a simple hill-climbing search. Otherwise the relative cost measure looses performance. The performance of all other heuristics variates slightly, some end up better others worse than hill-climbing.

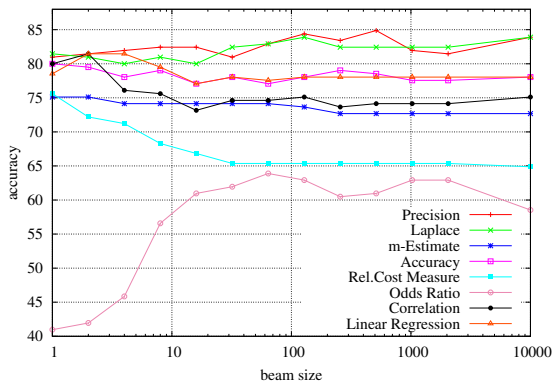
Among all plots, the one of "breast-w-d" shows the strongest fluctuations. However, this is partly due to the different scale: note that the y-axis only displays about 2% of change in accuracy. Another interesting plot is the one of "primary-tumor". The variations with all different beams are not that strong but nearly all heuristics show a clear peak when changing the beam search to the exhaustive search. This data set has 17 nominal attributes which have 43 values in total and 22 classes. Thus the search space for this data set is rather large. In several datasets, we can also see that, as we have already observed above, the performance of *Precision* and *Laplace* become increasingly similar with larger beam sizes.

### 6.4 Searching for single rules

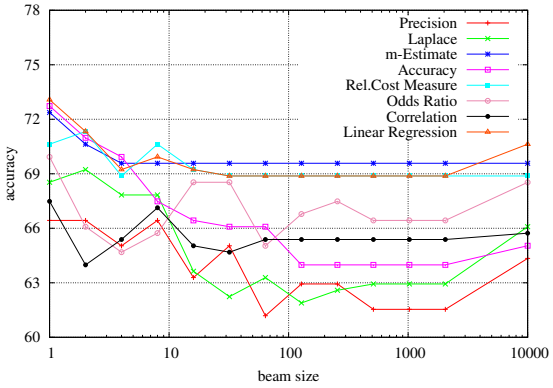
Table 3 displays results for a learning algorithm that only induces one rule per class. The purpose of these experiments is to observe the difference of the behavior not on entire theories, but on single rules.

For each heuristic, Table 3 compares the performance of the hill-climbing search (first line) to the performance of exhaustive search (second line). The accuracies are obtained by

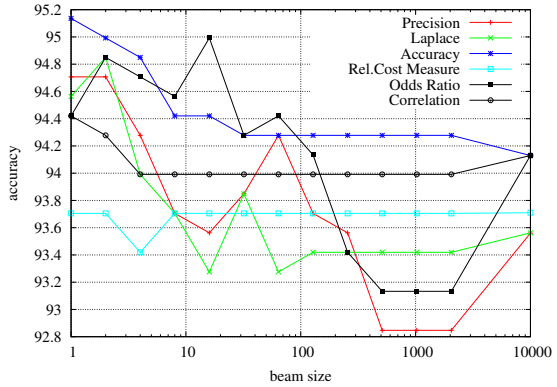
Figure 4: Beam size vs. accuracy for single data sets



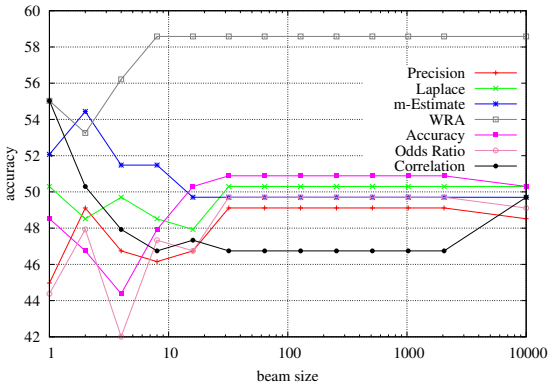

(a) autos-d



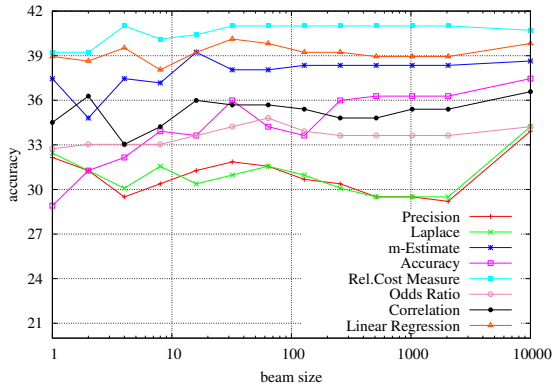

(b) breast-cancer



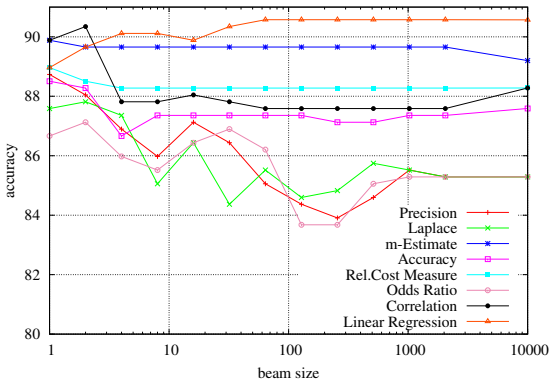

(c) breast-w-d




(d) monk2




(e) primary-tumor




(f) vote-1

using the rules in the same way as a complete model. Invariably, for all heuristics, exhaustive search finds rules that are longer, but more accurate. This can also be observed if we look at the average number of conditions per rule in the scenario that learns complete theories. Although the theories generally tend to become smaller in size, the individual rules tend to

Table 3: Accuracy and Theory Size when learning a single rule

| Heuristic | beam | accuracy in % | # conditions |
|---|---|---|---|
| precision | 1 | 64.67 | 6.82 |
| | exh. | 68.55 | 9.59 |
| laplace | 1 | 66.13 | 6.86 |
| | exh. | 68.78 | 9.64 |
| *m*-estimate | 1 | 70.87 | 7.23 |
| | exh. | 71.32 | 7.36 |
| WRA | 1 | 68.14 | 3.23 |
| | exh. | 68.81 | 3.5 |
| Accuracy | 1 | 72.50 | 6.64 |
| | exh. | 73.45 | 9.55 |
| RCM | 1 | 72.31 | 5.64 |
| | exh. | 72.69 | 6.00 |
| Odds Ratio | 1 | 63.99 | 4.05 |
| | exh. | 65.57 | 5.05 |
| Corr. | 1 | 72.41 | 7.14 |
| | exh. | 72.72 | 8.95 |
| Meta-L. | 1 | 70.38 | 7.27 |
| | exh. | 70.30 | 7.50 |

become longer with increasing beam sizes. This is because the FindBestRule procedure is able to find more complex rules that show a tendency to cover more examples than those induced by the simple hill-climbing search. As a result, the learned rules often become longer but cover more examples, so that fewer rules are necessary in total to cover all positive examples.

If we compare the results with those where complete models are learned in terms of accuracy, it is rather surprising that they achieve such high values. Most of the heuristics only lack about 10% accuracy in average, *Accuracy* and *Correlation* only decrease their performance by approximately 5%. Comparing their model size, both *Precision* and *Laplace* are able to achieve the described accuracies with theories that are about 7 times smaller than the complete theories, for the other the sizes are reduced to approximately half the size.

### 6.5 Covering algorithms and deep search

The previous discussions of the different results revealed some problems and inconvenient observations. In some cases a deeper search leads to theories that have a lower training set accuracy as their hill-climbing or low beam size variants. At first this seems to be very strange because usually a better search should induce models that describe the training data more accurate. This results from the ability of the FindBestRule procedure to find rules that cover more examples but remain at a high quality level. However, this holds only for *Precision* because this heuristic ensures that a rule is only added if it covers no negative examples, if there is any (which usually is assured for datasets without inconsistencies).

All of the other heuristics do not optimize the consistency of a rule in that excessive way. Thus, most of the found rules do cover negative examples and still are able to receive a high evaluation.

With a deeper search the probability of finding a rule that still satisfies the criterion of achieving a higher evaluation than the current best rule but covers a significantly higher number of negative examples (and positives, of course) increases. In general Covering algorithms are not able to exclude negative examples, that are covered once in further conquering steps. Thus, if in an early loop of the main covering procedure a rule is added, that, for example, covers more negative examples than the complete theory learned with hill-climbing, the accuracy of the theory cannot be improved any more. Therefore the differences from theories learned by hill-climbing and exhaustive search mostly lie in the smaller number of rules where those for the latter usually contain a higher number of conditions (as often observed in Figures 3 and 4 and Table 3).

Following from these observations we are able to draw another conclusion. All of the used heuristics are designed to induce a single rule that, in some certain sense, optimizes a quality criterion. This criterion differs among the heuristics used in this paper. But all of them have in common that they not aim to induce a rule that optimizes the whole theory. A known problem of Covering algorithms is that the rule is not learned in context to the incomplete theory that is learned so far. This effect also manifests in the stopping criterion described in Section 2.2 which states that a rule is only added to the theory if it covers more positives than negatives. If assumed that the best rule that could be found on the data is one that does not fulfil this criterion it would not be added although no better rule can be found. However, if we assume that the same heuristic is used to induce a single rule which evaluates the whole theory, a constant gain in performance should be measurable. As seen in Figure 4e this actually holds for the heuristic *Accuracy*. We are sure that if a rule is evaluated in context of the theory that is learned so far it would profit very much when using deeper search. On the contrary, if the theory size is of no concern, hill-climbing methods will work better because the induced rules do not cover as many negatives as with an exhaustive search.

## 7. Conclusion

The main conclusion that we draw from the experiments reported in this paper is that the over-searching phenomenon is highly dependent on the search heuristic. The performance of heuristics that have a bad performance with greedy search (such as *Odds Ratio* or *Precision*) can be considerably improved with exhaustive search, whereas heuristics that are optimized for greedy search will lose performance when used for guiding exhaustive search. This is, maybe most obvious when we compare the performance of *Precision* and *Laplace*, which differ greatly in their performance in a hill-climbing search, but perform almost identically when used in exhaustive search.

In general, we found that heuristics that perform badly with hill-climbing search (such as *Precision* or *Odds Ratio*) could be improved with exhaustive search, whereas the performance of heuristics that performed well for hill-climbing decreased with increased beam widths. In particular, the three heuristics that we had optimized for hill-climbing search in previous work, performed best, and their performance decreased steadily and substantially

when used with increasing beam widths, as is predicted by the over-searching phenomenon. In these cases, they learned much fewer, but substantially longer rules. Among the heuristics that we looked at, there was no counter-part that performed as well in exhaustive search than these heuristics performed in hill-climbing.

However, we would be careful not to interpret this as conclusive evidence that exhaustive search is detrimental and will necessarily lead to the over-searching phenomenon. Instead, we attribute this result to the fact that search heuristics for rule learning algorithms have to address several goals simultaneously: on the one hand, they have to estimate a rule's predictive quality, on the other hand, they have to evaluate the rule's potential for being refined into a rule that has a high predictive quality. However, with increasing search depth, the importance of the latter point decreases, because the chances that high-quality rules will be found without guidance of the search increase.

Thus, we think that good heuristics for exhaustive search have different requirements than good heuristics for hill-climbing search. Most of the efforts in inductive rule learning have been devoted only to the latter problem, whereas we would argue that finding a suitable metric for exhaustive rule induction is still an open problem. We plan to address this in future way in a similar study as we have previously performed for hill-climbing search (Janssen & Fürnkranz, 2008). As soon as we have identified a good evaluation metric for predictive rule learning, we can address the second step, which is to clearly separate the search heuristic and the rule evaluation metric in inductive rule learning algorithms.

### Acknowledgments

This research was supported by the *German Science Foundation (DFG)* under grant no. FU 580/2-1.

## References

Cohen, W. W. (1995). Fast Effective Rule Induction. In Prieditis, A., & Russell, S. (Eds.), *Proceedings of the 12th International Conference on Machine Learning*, pp. 115–123, Tahoe City, CA. Morgan Kaufmann.

Domingos, P. (1999). The role of occam's razor in knowledge discovery. *Data Mining and Knowledge Discovery*, *3*(4), 409–425.

Fürnkranz, J. (1999). Separate-and-Conquer Rule Learning. *Artificial Intelligence Review*, *13*(1), 3–54.

Janssen, F., & Fürnkranz, J. (2007). On meta-learning rule learning heuristics. In *Proceedings of the 7th IEEE Conference on Data Mining (ICDM-07)*, pp. 529–534.

Janssen, F., & Fürnkranz, J. (2008). An empirical quest for optimal rule learning heuristics. Tech. rep. TUD-KE-2008-01, Technische Universität Darmstadt, Knowledge Engineering Group. `http://www.ke.informatik.tu-darmstadt.de/publications/reports/tud-ke-2008-01.pdf`.

Lavrač, N., Flach, P., & Zupan, B. (1999). Rule evaluation measures: A unifying view. In Džeroski, S., & Flach, P. (Eds.), *Proceedings of the 9th International Workshop on Inductive Logic Programming (ILP-99)*, pp. 174–185. Springer-Verlag.

Lavrač, N., Flach, P. A., Kasek, B., & Todorovski, L. (2002). Rule induction for subgroup discovery with cn2-sd. In Bohanec, M., Kasek, B., Lavrac, N., & Mladenic, D. (Eds.), *ECML/PKDD'02 workshop on Integration and Collaboration Aspects of Data Mining, Decision Support and Meta-Learning*, pp. 77–87. University of Helsinki.

Michalski, R. S. (1969). On the Quasi-Minimal Solution of the Covering Problem. In *Proceedings of the 5th International Symposium on Information Processing (FCIP-69)*, Vol. A3 (Switching Circuits), pp. 125–128, Bled, Yugoslavia.

Murthy, S. K., & Salzberg, S. (1995). Lookahead and pathology in decision tree induction. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, pp. 1025–1031, Montreal, Canada. Morgan Kaufmann.

Newman, D., Blake, C., Hettich, S., & Merz, C. (1998). UCI Repository of Machine Learning databases..

Quinlan, J. R., & Cameron-Jones, R. M. (1995). Oversearching and layered search in empirical learning. In *IJCAI*, pp. 1019–1024.

Todorovski, L., Flach, P., & Lavrač, N. (2000). Predictive performance of weighted relative accuracy. In Zighed, D., Komorowski, J., & Zytkow, J. (Eds.), *Proceedings of the 4th European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD-2000)*, pp. 255–264, Lyon, France. Springer-Verlag.

Webb, G. I. (1995). OPUS: An efficient admissible algorithm for unordered search. *Journal of Artificial Intelligence Research*, *3*, 431–465.

Witten, I. H., & Frank, E. (2005). *Data Mining — Practical Machine Learning Tools and Techniques with Java Implementations* (2nd edition). Morgan Kaufmann Publishers.