UNIVERSITEIT GENT

FACULTEIT ECONOMIE EN BEDRIJFSKUNDE

TWEEKERKENSTRAAT 2
B-9000 GENT
Tel. : 32 - (0)9 – 264.34.61
Fax. : 32 - (0)9 – 264.35.92

WORKING PAPER

To Tune or not to Tune: Rule Evaluation for Metaheuristic-based Sequential Covering Algorithms

Minnaert Bart[*]

Martens David[†]

De Backer Manu[‡]

Baesens Bart[§]

January 2012

2012/769

[*] Affiliated researcher at Dept. Management Information Science and Operations Management, Faculty of Economics and Business Administration, Ghent University, Belgium.
Department of Business Administration and Public Management, University College Ghent, Belgium.
Faculty of Applied Economics, University of Antwerp, Belgium.
[†] Faculty of Applied Economics, University of Antwerp, Belgium.
[‡] Department of Business Administration and Public Management, University College Ghent, Belgium.
Faculty of Applied Economics, University of Antwerp, Belgium.
Department of Decision Sciences Information Management, K.U.Leuven, Belgium.
[§] Department of Decision Sciences Information Management, K.U.Leuven, Belgium.

D/2012/7012/02

To Tune or not to Tune: Rule Evaluation for Metaheuristic-based Sequential Covering Algorithms

Bart Minnaert · David Martens · Manu De Backer · Bart Baesens

Abstract While many papers propose innovative methods for constructing individual rules in separate-and-conquer rule learning algorithms, comparatively few study the heuristic rule evaluation functions used in these algorithms to ensure that the selected rules combine into a good rule set. Underestimating the impact of this component has led to suboptimal design choices in many algorithms. The main goal of this paper is to demonstrate the importance of heuristic rule evaluation functions by improving existing rule induction techniques and to provide guidelines for algorithm designers. We first select optimal heuristic rule learning functions for several metaheuristic-based algorithms and empirically compare the resulting heuristics across algorithms. This results in large and significant improvements of the predictive accuracy for two techniques. We find that despite the absence of a global optimal choice for all algorithms, good default choices seem to exist for families of algorithms. A near-optimal selection can thus be found for new algorithms with minor experimental tuning. A major contribution is made towards balancing a model's predictive accuracy with its comprehensibility, as the parametrized heuristics offer an unmatched flexibility when it comes to setting the trade-off between accuracy and comprehensibility.

Bart Minnaert (E-mail: Bart.Minnaert@Ugent.be)
Department of Business Administration and Public Management, University College Ghent, Ghent University, Belgium
Faculty of Applied Economics, University of Antwerp, Belgium

David Martens (E-mail: David.Martens@ua.ac.be)
Faculty of Applied Economics, University of Antwerp, Belgium

Manu De Backer (E-mail: Manu.DeBacker@Hogent.be)
Department of Business Administration and Public Management, University College Ghent, Ghent University, Belgium
Faculty of Applied Economics, University of Antwerp, Belgium
Department of Decision Sciences Information Management, K.U.Leuven, Belgium

Bart Baesens (E-mail: Bart.Baesens@econ.kuleuven.be)
Department of Decision Sciences Information Management, K.U.Leuven, Belgium

Keywords Classification · Rule Induction · Heuristics · Rule Evaluation · Sequential Covering

1 Introduction

Rule sets are a popular modeling technique for the classification task. Many techniques have been developed for the induction of first-order rules from a set of examples to predict a nominal target variable on unseen data. While sophisticated non-linear techniques such as support vector machines (SVM) and artificial neural networks (ANN) offer high performance (Baesens et al, 2003b; Lessmann et al, 2008; Van Gestel et al, 2004), rule sets can provide other advantages. In some important domains, such as medical and financial applications, there is a need for validation by domain experts (Baesens et al, 2003a; Pazzani et al, 2001; Verbeke et al, 2011). Models in these domains thus need to be comprehensible. The non-linear techniques result in black box models, while rule lists can provide more comprehensibility. Rule induction algorithms can be applied in these domains or can be combined with non-linear techniques to build a comprehensible rule set (Andrews et al, 1995; Martens et al, 2007a).

Several strategies have been used in these rule induction algorithms to search for good rule sets (Tan et al, 2005). A direct approach would be to optimize an entire rule set, as seen in several genetic algorithms (Freitas, 2003). An indirect approach consists of extracting rule sets from decision trees (Quinlan, 1993). However, the central problem we study in this paper occurs in algorithms that use the separate-and-conquer or sequential covering strategy to build ordered rule sets (Fürnkranz, 1999). In this strategy, individual rules are iteratively induced and then appended to the rule set. It reduces the bigger problem of finding the best rule set to the lesser problem of finding the best rule, which is still a complex task.

A key difficulty arises in the separate-and-conquer strategy because the goals - e.g. accuracy, comprehensibility - are defined with respect to the resulting rule set, while the search procedure evaluates and induces each rule individually. The heuristic rule evaluation function can only define the objectives for the search procedure. It thus has to define these objectives in such a way that the overall goals are maximized. However, exactly how the heuristic rule evaluation function affects the overall performance of the rule set is not clear.

Almost all new algorithms that use this separate-and-conquer strategy will focus on novel solutions to finding the best rule. However, the important problem of defining what the best rule is, is still unresolved and receives a lot less attention from researchers (Witten and Frank, 2005). In general, a heuristic rule evaluation function is used that defines the quality of a rule in a way that the algorithm will find the best rule set. Several such heuristic rule evaluation functions have been proposed by various authors over the years. This is usually as part of a new algorithm with some small-scale experimental com-

parison with one or more alternatives. For example, the PSO/ACO2 algorithm introduced a heuristic[1] that was claimed to perform better than the heuristic implemented in PSO/ACO (Holden and Freitas, 2008). Fürnkranz and Flach (2005) consolidated most of the earlier research in a theoretical analysis that proves the equivalence of several proposed heuristics. A recent study by Janssen and Fürnkranz (2010) is the first large-scale experimental benchmarking study of heuristic rule evaluation functions for separate-and-conquer rule induction. In the study, several heuristics - including parametric heuristics - are evaluated in the CN2 algorithm (Clark and Niblett, 1989) with the goal of finding the optimal heuristic for this task. Recently, a different view was taken by Salama and Abdelbar (2011) in a small scale empirical study of heuristics on μAntMiner. While Janssen and Fürnkranz (2010) focus on optimal accuracy, Salama and Abdelbar (2011) reason that comprehensibility is important as well and investigate the trade-off between both. This is a valid concern as rule sets are often used when comprehensibility is required. However, the study only investigates several non-parametrized heuristics on a single algorithm. Janssen and Fürnkranz (2010) further observed that the relative performance of heuristics may be algorithm-dependent. This point of view was further argued by Janssen and Fürnkranz (2009) in a study on the over-searching phenomenon, which reports that the relative performance of heuristics depends on the exhaustiveness of the search.

This interplay between the heuristic rule evaluation function and the characteristics of the algorithm still requires further investigation. We examine this interplay in algorithms that use metaheuristic search to optimize individual rules. Metaheuristics are high level strategies for iteratively finding high quality solutions given a search space and an objective function. First we attempt to improve several metaheuristic-based algorithms by selecting a heuristic rule evaluation function that is more optimal with respect to the predictive accuracy. As the heuristic rule evaluation function is a hyper component that is part of many algorithms, we further compare the resulting functions in order to provide guidelines for future research. Lastly, we investigate the usefulness of parametrized heuristics to select a suitable trade-off between accuracy and comprehensibility. To the best of our knowledge, this is the first large scale empirical study of heuristic rule learning functions on multiple algorithms.

The rest of this paper is organized as follows: in Section 2, the background regarding sequential covering, heuristic rule evaluation functions and the algorithms used in this work is provided; the methodology and our experimental setup is presented in detail in Section 3; Section 4 contains the empirical results of the experiments; in Section 5, we discuss the implications of the experimental results and in Section 6 we summarize the main findings of this work and present guidelines for incorporating this work into new or existing algorithms.

[1] Several terms are encountered in the literature that can refer to heuristic rule evaluation functions. Alternative terminology includes 'rule learning heuristic', '(rule) evaluation function', 'fitness function' and '(rule) quality measure'. For the sake of brevity, the term 'heuristic' is used in this text where applicable.

2 Sequential Covering

A popular approach to rule induction is the separate-and-conquer or sequential covering strategy. This strategy constructs an ordered rule set by iteratively selecting rules and is used by many well-known algorithms such as RIPPER (Cohen, 1995) and CN2 (Clark and Niblett, 1989). The key idea behind this strategy is that a rule set that contains only good rules is likely a good rule set in turn. Under this assumption, the algorithm reduces the problem of finding the best rule set to the problem of finding the best rule. As we will explain in the next paragraphs, a good heuristic rule evaluation function is a key component of this strategy that ensures the extracted rules form a good rule set.

Algorithm 1 contains an overview of the required components of a sequential covering algorithm. Starting with the complete training set, a first rule is extracted that accurately covers a large part of the training set. The rule is added to the empty rule list and all covered examples are removed from the training set. The algorithm then continues to find rules for the uncovered examples, removing all examples covered in each step, until a stopping criterion is reached. All remaining examples are then covered by a default rule that assigns the majority class to all examples. Each algorithm using this strategy has to implement the procedures FINDBESTRULE and STOPCRITERION. Other procedures can be added as well, for example RIPPER introduces a postprocessing step to further optimize the rules, but they are not key to the sequential covering strategy.

The first component of this algorithm is the stopping criterion. This component determines when the rule set is complete and, ideally, cannot be further improved by adding another rule, except for the default rule. A commonly used stopping criterion is a lower bound on the number of uncovered examples. In that case, rule induction is stopped when only a certain number or percentage of examples is left in the training set. Alternative methods exist, for example the early stopping criterion in AntMiner+ that tracks the performance of the rule set on a separate validation set (Martens et al, 2007b).

The second component (FINDBESTRULE) is a procedure to extract a good rule from a set of examples. Many different methods have been implemented over the years to perform this complex task. In general, this is an optimization problem of the quality of a rule over the space of all first order rules. An exhaustive search of this space is infeasible for all but trivial problems, so most algorithms propose a heuristic search procedure. One class of traditional algorithms explicitly construct a rule by iteratively refining the rule through specialization or generalization (Fürnkranz, 1999). Another class of algorithms uses metaheuristic search to find a good rule. This class contains genetic algorithms (GA), algorithms based on ant colony optimization (ACO), particle swarm optimization (PSO) or hybrid algorithms.

The metaheuristic-based algorithms need to instantiate the components of the metaheuristic search to the problem at hand - finding the best rule. Furthermore they require an explicit function that defines the quality of a rule

Algorithm 1 Sequential Covering

$RuleSet = \emptyset$;
repeat
  $BestRule = FINDBESTRULE(Examples)$
  $RuleSet = Ruleset \cup BestRule$
  $Examples = Examples - Covered(BestRule)$
until STOPCRITERION
$RuleSet = RuleSet \cup DefaultRule$
return $RuleSet$

to be optimized - the heuristic rule evaluation function. The algorithm thus selects a rule with high values for the search space defined by the heuristic rule evaluation function. In contrast, the construction algorithms employ a search heuristic to further refine a solution in a local search procedure. Some will employ a heuristic evaluation function to select the best refinement - e.g. adding a condition - at each step and thus explicitely define a search space that is traversed using methods such as hill climbing. Other algorithms however have no explicit function of the quality of a rule and measure the quality of the refinement relative to the rule being refined using a more general gain heuristic. These gain heuristics can not be applied in this study as it concerns metaheuristic-based algorithms. These require a heuristic rule evaluation function that maps a single rule to a quality or fitness value. We will first give an overview of heuristic rule evaluation functions. We conclude this section with a brief description of the five algorithms that we will use in this study.

2.1 Heuristic rule evaluation functions

In general, heuristic rule evaluation functions map a rule R to a fitness value $h(R)$ with higher values corresponding to better rules. This fitness function is usually specified as a function $h(p, n, P, N)$ of several basic metrics: p and n refer to respectively the number of correctly and incorrectly covered examples, also referred to as true positives and false positives. P is the total number of examples of the target class remaining in the training set, while N is the total number of examples belonging to other classes. Both P and N are often used to normalize the function and/or to take the class distributions into account. Other basic metrics are used as well, for example the number of terms in a rule can be used to introduce a bias towards rules with fewer terms, but the majority of algorithms only uses the above metrics (Fürnkranz, 1999).

The search procedure searches the rule space for rules with high values for the fitness function. This use as an evaluation function is most explicit in algorithms that use metaheuristic search. In more traditional construction algorithms, the function has a dual purpose. In these algorithms, it is also used as a search heuristic in a hill climbing or beam search. The function has to give high fitness values both to good rules and to rules that can be refined into good rules. This is usually done through specialization of a rule by adding extra terms. The results of Janssen and Fürnkranz (2009) suggest

that heuristics that work well in metaheuristic search do not necessarily work well when used in a construction algorithm.

The question remains how to describe with these metrics what a good rule is. Rules are better if they cover more true positives p and/or less false positives n. However, generalizing a rule so that it covers more true positives often leads to covering more false positives as well. A good trade-off is needed to satisfy both conflicting goals. A naive approach consists of applying the objective of maximized precision, $p/(p+n)$, of the rule set to each rule individually. The precision is however easily maximized by rules that cover only a single training example. This approach thus leads to finding rules that cover few examples. These rules typically do not generalize well and as a result the precision of the rule set on test data will be low due to overfitting. On the other hand, the (positive) coverage, $p/(P+N)$, looks exclusively at the number of correctly covered examples and can easily be maximized by a rule that covers all examples. Several heuristics make an explicit trade-off between precision and coverage. On one hand, weighing coverage too heavily results in rules with low precision on both training and test set. On the other hand, weighing coverage not heavily enough results in rules with a precision that is high on the training set, but low on the test set.

The heuristics used in practice are often not uniquely used for this task. Several tasks require the use of an evaluation function or heuristic that balances true positives and false positives. This type of measure is used in for example unordered rule induction (An and Cercone, 2000), association analysis (Tan et al, 2002), contrast set, emerging pattern, subgroup mining (Novak et al, 2009) and even for the evaluation of classifiers (Witten and Frank, 2005). Heuristics are often borrowed from these related tasks. In the case of separate-and-conquer rule learning, precision should be weighted more heavily than coverage (Janssen and Fürnkranz, 2009). This is probably due to an asymmetry introduced by separate-and-conquer. When a rule is appended to the rule set, the examples are permanently covered. The false positives can thus no longer be corrected by the algorithm.[2] However, the false negatives can still be covered by subsequent rules. Heuristics that perform well in other domains do not necessarily do so for separate-and-conquer rule learning due to this asymmetry. In (Janssen and Fürnkranz, 2009) for example, a parameter value that favors coverage was taken for the Klösgen measure from research on subgroup discovery (Wrobel, 1997). This parameter setting turned out to have low performance for separate-and-conquer. Several algorithms do not take this asymmetry into account, for example AntMiner and AntMiner+ which are included in Section 2.2.

Many different heuristics are used in practice. A 1999 review of separate-and-conquer rule learning (Fürnkranz, 1999) contains an overview of heuristics used in construction algorithms. A recent survey of swarm intelligence shows an overview of the heuristics used in this area, which comprises ant colony opti-

[2] The RIPPER algorithm is an exception as corrections can still be made in the post-processing step.

mization, particle swarm optimization and prey models (Martens et al, 2011). Eight separate-and-conquer algorithms were found in this domain of which six contain the same heuristic, being the multiplication of sensitivity and specificity. This heuristic was originally introduced in AntMiner (Parpinelli et al, 2002) and most likely propagated in this sub domain. A survey of genetic algorithms by Fernández et al (2010) shows that few algorithms in this class follow the separate-and-conquer strategy. Those that do propose their own new heuristic. Fürnkranz and Flach (2005, 2003) perform a theoretical analysis to demonstrate that several of the heuristics used in practice are equivalent to each other. They are often special cases of more general parametrized heuristics in which the trade-off can be set through a parameter. This study is limited to these parametrized heuristics as they allow a finely tuned trade-off and cover most basic heuristics. For example, Salama and Abdelbar (2011) employ 10 fixed heuristics of which 6 are equivalent to specific instances of the parametrized heuristics used in this study, including 3 out of 4 best performing heuristics. Table 1 contains an overview of this trade-off for the parametrized heuristics used in this study. We will discuss these heuristics in more detail in the next paragraphs. Heuristics implemented in algorithms used in this study are discussed in Section 2.2.

2.1.1 Klösgen measure

$$f_K(\omega) = (\frac{p+n}{P+N})^{\omega} \cdot (\frac{p}{p+n} - \frac{P}{P+N}) \tag{1}$$

The Klösgen measure is defined by (1). It multiplicatively trades off full coverage and precision. An interesting feature is that the precision is corrected for the class distribution. The Klösgen parameter ω controls the weight assigned to the coverage. For $\omega = 0$, the Klösgen measure equals precision. At $\omega = 1$, the Klösgen heuristic is equivalent to the weighted relative accuracy, $p/P - n/N$. This setting balances the true positive rate and false positive rate. For higher values of ω, coverage dominates the equation. In the limit $\omega \to \infty$, the Klösgen measure equals full coverage. This measure was first proposed in (Klösgen, 1992) and is used in subgroup discovery. It was recently introduced for separate-and-conquer rule induction with good results (Janssen and Fürnkranz, 2009). There it was shown that $\omega < 1$ is the optimal region for separate-and-conquer rule induction.

Table 1 Precision vs. coverage trade-off for heuristics

Heuristic	Precision	Balanced	Coverage
Klösgen	$\omega = 0$	$\omega = 1$	$\omega \to \infty$
F-measure	$\beta = 0$	$\beta = 1$	$\beta \to \infty$
Relative cost	$c = 0$	$c = 0.5$	$c = 1$
m-estimate	$m = 0$	$m \to \infty$	

2.1.2 F-measure

$$f_F(\beta) = \frac{(\beta^2+1)\cdot\frac{p}{p+n}\cdot\frac{p}{P}}{\beta^2\cdot\frac{p}{p+n}+\frac{p}{P}} \tag{2}$$

$$= (\beta^2+1)\cdot\frac{p}{p+n+\beta^2 P} \tag{3}$$

Equation (2) shows the F-measure. It is the weighed harmonic mean of precision and coverage. For $\beta = 0$, the F-measure equals precision. Precision and recall are weighed equally for $\beta = 1$. In the limit $\beta \to \infty$, the F-measure becomes recall. It was originally proposed as an evaluation measure in information retrieval, where the parameter indicates that the user attaches β times as much importance to recall as precision (van Rijsbergen, 1979).

Equation (2) can be rewritten as (3), in which the constant factor (β^2+1) can be ignored for rule evaluation. The F-measure can thus be interpreted as a generalization of the Laplace-corrected precision given by (4), of which PSO/ACO2 implements a variant. The assumed a-priori coverage of a rule is $\beta^2 P$. It is also related to the m-estimate, which is described further on.

$$f_{Laplace} = \frac{p+1}{p+n+2} \tag{4}$$

2.1.3 Relative cost measure

$$f_{RCM}(c) = c\cdot\frac{p}{P} - (1-c)\cdot\frac{n}{N} \tag{5}$$

The relative cost measure as used by Janssen and Fürnkranz (2010) is defined by (5). It balances the true positive rate and false positive rate through a cost parameter c. For $c = 0$, the relative cost measure exclusively punishes the false positive rate, which leads to smaller high precision rules. On the other hand, it purely rewards the true positive rate for $c = 1$, which leads to high coverage rules. A balance is found for $c = 0.5$, for which the relative cost measure equals the weighted relative accuracy as with the Klösgen measure.

2.1.4 m-estimate

$$f_m(m) = \frac{p + m\cdot\frac{P}{P+N}}{p+n+m} \tag{6}$$

The m-estimate is defined by (6). It is an extension to the well-known Laplace corrected precision. Where the Laplace correction assumes an a priori coverage of one correctly and one incorrectly classified example, the m-estimate assumes an a priori coverage of m examples with a distribution equal to the class distribution (Cestnik, 1990). For $m = 0$, an a priori coverage of zero

leads to precision. Increasing m weighs coverage more heavily. It was shown by Fürnkranz and Flach (2003) that the m-estimate converges towards the weighted relative accuracy in the limit $m \to \infty$.

The interpretation of the parameter m as the number of a priori covered examples implies that the value of m scales with the size of the dataset. Indeed, artificially increasing the size of the dataset by duplicating all examples with a factor α, is equivalent to setting the parameter to m/α. Larger datasets typically require higher values for m and thus some minor experimentation was done with variants that might scale well, but surprisingly the proposed variants had a lower performance. We speculate this did not work because the number of rules typically increases too with the dataset size, making the required increase in m non-linear.

2.2 Algorithms

The many separate-and-conquer rule mining algorithms implement a wide variety of ideas for extracting good rules. In this work, we focus on algorithms that use a metaheuristic search to perform this task. We included two ACO-based algorithms, one PSO/ACO hybrid algorithm and one genetic algorithm. We further included the java implementation of the well-known construction algorithm RIPPER to discuss potential differences for construction algorithms and metaheuristic-based algorithms with regards to the rule evaluation function. To summarize, the five algorithms included are AntMiner, AntMiner+, PSO/ACO2, HIDER and RIPPER. This selection is based on available open source implementations, reported good performance in literature and includes at least one leading algorithm per class. Furthermore, closely related algorithms, such as the AntMiner variants, are left out of the selection as they are likely to produce very similar results.

Before we give a brief summary of these algorithms, we will point out a few general characteristics of these algorithms that may influence the performance of the heuristic rule evaluation functions. An overview of these characteristics is given in Table 2. As already discussed in Section 2.1, heuristics in construction algorithms also evaluate the ability of partial rules to be refined into good rules later on. As these good partial rules typically have a higher coverage, this can influence the heuristic. A second general algorithmic factor is concerned with how the fitness value is processed. Most algorithms will use this fitness value to establish an order relation between the rules. However some algorithms

Table 2 Algorithmic components influencing rule evaluation

Algorithm	Rule evaluation function	Partial rules	Exact values	Pruning	Min_cases
AntMiner+	precision + coverage		✓	✓	
AntMiner	sensitivity × specificity		✓	✓	✓
PSO/ACO2	laplace corrected precision		✓	✓	✓
HIDER	accuracy + geometric coverage		✓		
RIPPER	information gain	✓		✓	

also process the exact values, most notably the ACO-based algorithms. These algorithms may find it harder to leverage upon very small differences in fitness between rules.[3] Functions that are too flat may have lower performance in these algorithms. A third factor is the presence of overfitting-avoidance measures such as pruning. Pruning typically guides the search towards higher coverage rules. Another measure used in several algorithms is a lower bound on the number of true positives covered. These measures can prevent some of the negative impact due to overfitting of heuristics focussing on precision.

2.2.1 AntMiner

AntMiner is the first application of ant colony optimization for the classification task (Parpinelli et al, 2002, 2001).[4] It implements the separate-and-conquer strategy with the standard stopping criterion of appending rules until the number of remaining examples falls below a certain level, specified by $max\_uncovered\_cases$. Individual rules are mined through the application of ACO.

When applying the ACO metaheuristic, first an environment in which the ants operate needs to be defined in a way that when the ants move, they incrementally construct a solution to the problem at hand, in this case the classification problem. The AntMiner environment is defined as a directed graph, where for each variable there are as many nodes as there are values for that variable. Bidirectional edges exist between all nodes from different variables. This choice of environment allows the ants to choose which variable to add next, but limits the algorithm to nominal variables only.

AntMiner implements the Ant System variant of the ACO metaheuristic (Dorigo et al, 1996). As an ant traverses the environment, it constructs a rule. This rule is then evaluated and pheromone is deposited on the edges visited by the ant proportional to the fitness value of the rule. Pheromone is also evaporated at this time through the normalization of the pheromone matrix. This allows the ants to 'forget' bad paths. When ants traverse the environment, they are more likely to select edges that have high pheromone levels. New ants traverse the environment until either the threshold no_of_ants is reached or $no\_rules\_converg$ consequent ants construct the same rule. Furthermore, AntMiner employs pruning as a hill climbing local search before updating the pheromone. It also implements a $min\_cases\_per\_rule$ parameter, which specifies the minimal coverage of a rule. Both measures guide the search towards higher coverage rules.

$$f_{\text{AntMiner}} = \frac{p}{P} \cdot \frac{N - n}{N} \tag{7}$$

[3] Negative fitness values are also not allowed in these algorithms. For this reason, all parametrized heuristics described in the previous section were incremented with a constant to bring the lowest possible value at precisely zero.

[4] AntMiner is available at http://sourceforge.net/projects/guiantminer/

The heuristic used is the sensitivity multiplied with specificity, given by (7). This heuristic balances the sensitivity and specificity measures that are often used to evaluate classifiers in the medical domain (Witten and Frank, 2005). It was originally proposed by Lopes et al (1997) as a fitness function for a fuzzy logic classifier in which the weights were optimized through a genetic algorithm. This heuristic was then introduced for heuristic rule evaluation function in a separate-and-conquer setting in AntMiner. A recent survey of Swarm intelligence algorithms shows that the use of this heuristic propagated to all but two algorithms in this domain (Martens et al, 2011). This is not surprising as most ACO-based algorithms are in fact based on AntMiner. Salama and Abdelbar (2011) noted that several other non-parametrized heuristics have a higher performance on both predictive accuracy and comprehensibility on μAntMiner.

2.2.2 AntMiner+

A second ACO-based algorithm is AntMiner+ (Martens et al, 2007b).[5] While similar in name, it differs from AntMiner in several key areas such as the metaheuristic variant used and the environment used. Aside from the search procedure, AntMiner+ also introduces an early stopping criterion. For this criterion, the performance of the rule set is tracked on a separate validation set to determine when to stop the addition of new rules.

The environment is defined as a directed acyclic graph (DAG), so that the ants can choose their paths more effectively. Furthermore, to allow for interval rules, the construction graph additionally exploits the difference between nominal and ordinal variables: each nominal variable has one node group (with the inclusion of a dummy vertex indicating the variable does not occur in the rule), but for the ordinal variables however, two node groups are built to allow for intervals to be chosen by the ants. The first node group corresponds to the lower bound of the interval and the second node group determines the upper bound.

AntMiner+ implements the better performing MAX-MIN Ant System (Stützle and Holger, 2000). *no_ants* ants traverse the environment in each iteration, but only the best rule is pruned and reinforced through a pheromone update. This allows for a better exploitation of the best solution found. The range of possible pheromone trails is limited to an interval $[\tau_{min}, \tau_{max}]$ so as to avoid early stagnation of the search. The initial pheromone value of each trail is set at τ_{max}. This determines a higher exploration at the beginning of the algorithm.

$$f_{\text{AntMiner+}} = \frac{p}{p+n} + \frac{p}{P+N} \tag{8}$$

AntMiner+ also proposes a new heuristic, which is the sum of the confidence - synonymous for precision - and (positive) coverage given by (8). This heuristic thus balances precision and coverage directly in an additive way. A

[5] AntMiner+ is available at http://www.antminerplus.com

potential problem with this approach is associated with classes that occur frequently in the dataset. A rule that assigns all examples to the majority class, can simultaneously maximize coverage and maintain a relatively high precision. To avoid this, AntMiner+ excludes rules for the majority class from the search.

2.2.3 PSO/ACO2

PSO/ACO2 is a hybrid algorithm that combines ant colony optimization with particle swarm optimization (Holden and Freitas, 2008).[6] This combination is used to build rule-based classification models that can handle both nominal and continuous variables. While for PSO implementations nominal variables need to be encoded as binary variables, and ACO implementations require discretization, PSO/ACO2 directly deals with both types of variables. This technique extends and refines the PSO/ACO algorithm by the same authors (Holden and Freitas, 2005). This algorithm also deviates substantially from the standard separate-and-conquer strategy. In fact, separate-and-conquer is run once for each class individually. All resulting rules are then ordered by fitness value and pruning is done to remove unnecessary terms and rules.

The selection of the best rule happens in two phases. First, a hybrid PSO/ACO search constructs a rule using only the nominal variables. Afterwards the continuous variables are used to improve this rule with a relatively standard PSO procedure. The hybrid algorithm contains several particles that each contain their own pheromone matrix. A random seed example is selected for each particle that is used to define the environment for that particle. Each particle is thus basically running a separate limited ACO search in which a particle can only choose whether it adds the terms from its seed example. However, the particles are arranged in a 2D-grid. They influence each other through the pheromone update rule, which takes into account the best rule found in the particle's neighborhood. The pheromone update is elitist in that it only reinforces based on the best rules found by the particle and its neighbors. This way, each particle stochastically constructs a rule based on its seed example and its associated pheromone matrix. If this rule has a higher fitness value than the best encountered by the particle, this new rule is remembered by the particle. Each particle then updates its pheromone matrix based on the best rules encountered by this particle and its neighbors. This is repeated $max\_iterations$ times, after which the best rule encountered is selected.

$$f_{\text{PSO/ACO2}} = \frac{p+1}{p+n+1} \tag{9}$$

PSO/ACO2 employs a version of the Laplace corrected precision as given by (9). Interestingly, the heuristic was changed from the $sensitivity \times specificity$ used in PSO/ACO. This change was instigated by the improved performance

[6] PSO/ACO2 is available at http://sourceforge.net/projects/psoaco2/

compared to the original heuristic. A problem with low coverage rules was in turn detected as this heuristic focuses heavily on precision. This in turn lead to the addition of a *min_cases* parameter. Unlike in AntMiner, rules with a lower coverage than this limit are still allowed, but their fitness value is severely reduced.

2.2.4 HIDER

HIDER is the most recently known genetic algorithm that takes a separate-and-conquer approach (Aguilar-Ruiz et al, 2003).[7] For this reason it was selected over the SIA (Venturini, 1993) algorithm. New rules are discovered until the number of remaining examples falls below a fraction of the initial size of the training set, controlled by the parameter efp.

An evolutionary algorithm searches for the best rule at each iteration. The population is composed of *population_size* rules. This population is initialized by rules that cover at least a single randomly selected example. *num_generations* generations are simulated before the best rule found is returned. The best individual is passed on to each generation, along with a set of individuals selected through the roulette wheel method.[8] A second set of individuals is the result of recombination after which a selection step reduces the population to its specified size using the roulette wheel method again. The most recent version of HIDER uses natural encoding, as described in detail by Aguilar-Ruiz et al (2007).

$$f_{\text{Hider}_1} = 2 \cdot (P + N - n) + p + V(R) \tag{10}$$

$$f_{\text{Hider}_2} = P + N - n + p + V(R) \tag{11}$$

$$f_{\text{Hider}_3} = P + N - \alpha \cdot n + p + V(R) \tag{12}$$

HIDER proposes several novel heuristic rule evaluation functions. The original paper uses the heuristic in (10) (Aguilar-Ruiz et al, 2003), but this was changed later to (11) in (Aguilar-Ruiz et al, 2007). However, closer inspection of the open source implementation shows that a penalty factor α was introduced as in (12). The default value of this pruning factor is $\alpha = 1$. In this work we use this setting, which is equal to the heuristic described in (11) as in (Aguilar-Ruiz et al, 2007). The HIDER heuristic is nearly equivalent to the accuracy measure, defined as $p - n$. A non-standard geometric coverage metric $V(r)$ is calculated as the proportion of the attribute space covered by the rule r. As $V(r) < 1$ holds, this metric functions as a tiebreaker for rules with equal accuracy in favor of rules that cover a larger section of the attribute space.

[7] An implementation of HIDER is available as part of the KEEL software project (Alcalá-Fdez et al, 2009)(Alcalá-Fdez et al, 2011) at http://www.keel.es/

[8] The probability of an individual to be selected is proportional to its fitness value. This method thus uses the exact values returned by the fitness function.

2.2.5 RIPPER

RIPPER is a construction algorithm that we include as the representative for this class because it is both well-known and highly accurate (Cohen, 1995).[9] While this research focuses mainly on metaheuristic-based algorithms, a construction algorithm is needed to investigate how differently these algorithms behave from the construction algorithms. A notable feature of RIPPER is the post-processing steps performed after the standard separate-and-conquer strategy. During post-processing, each rule is in turn selected and further optimized in the context of the existing rule set. This is repeated k times, with a default setting $k = 2$. The rule evaluation performed during this phase is fundamentally different from that in the initial rule set building phase because the performance of the rule in the classifier can now be measured. RIPPER also performs a separate-and-conquer loop for each class in the dataset. This is done starting from the least prevalent class and ending with the most prevalent class.

The creation of the best rule consists of two stages: a growing stage and a pruning stage that each use half of the remaining training data. RIPPER constructs a rule to predict the least prevalent class in the growing stage. Starting from the empty rule, it iteratively specializes the constructed rule by adding a refinement term to this rule. All possible refinement rules are evaluated to select the optimal one. This is repeated until no refinement is possible. The resulting rule is usually very specialized and has a high chance of overfitting. Thus in the second stage, the rule is generalized again through pruning using the remaining training data. Pruning is done by removing the last n terms, where n is chosen so that it maximizes the precision on these new training examples.

$$f_{\text{RIPPER}} = p \cdot (\log_2 \frac{p}{p+n} - \log_2 \frac{p'}{p'+n'}) \tag{13}$$

RIPPER evaluates each refinement of a rule by calculating the information gain. The information gain, defined by (13), was introduced in the FOIL algorithm (Quinlan, 1990). The statistics (p, n) of the refined rule are compared with those of the original rule, (p', n') in this heuristic. The fitness value assigned to a rule is thus relative to an incomplete rule, which makes it a gain heuristic. Depending on the path taken, the same rule can receive different fitness values. For this reason, gain heuristics can not be used in the metaheuristic-based algorithms described earlier.

3 Experimental Setup

The use of a parametric heuristic required a design for correct parameter setting. Tuning the parameter on each problem costs time and consumes data,

[9] We use the RIPPER implementation included in the Weka project (Hall et al, 2009) at http://www.cs.waikato.ac.nz/ml/weka/

which potentially reduces the performance gain of parameter tuning. Therefore optimal default values for the parametrized heuristics are computed for each algorithm individually in the first phase of the experiments as shown in the high-level overview of the experimental setup in Fig. 1. The tuning of the parameters is done using a grid search on a tuning bench of datasets as described in Section 3.2 and is similar to the method used by Janssen and Fürnkranz (2010) for the CN2 algorithm. The analyses of the trade-off between accuracy and comprehensibility described in Section 3.5 will use this data. The tuned heuristics are then evaluated on a validation bench of datasets with the primary goal of selecting the best heuristic for each algorithm and the secondary goal of evaluating the performance of each heuristic on each algorithm. In the last phase of the experiments, a cross-algorithm evaluation of the optimized heuristics is done to evaluate the performance gain from optimizing a heuristic for a specific algorithm.

3.1 Datasets

A tuning bench of 27 datasets[10] is used for the optimization of the heuristics and a validation bench of 23 datasets[11] is used for the evaluation. We use the same selection of datasets as in the CN2 study by Janssen and Fürnkranz (2010). However, several datasets were removed from the validation bench used in that study as they are not independent from the datasets in the tuning bench. Almost all datasets are available at the UC Irvine Machine Learning Repository (UCI) (Hettich and Bay, 1996).[12]

Because not all algorithms used in this study can handle missing values, a first preprocessing step creates versions of these datasets with no missing values. For attributes where the dataset description indicates that the value being missing is significant in itself, a separate value is added to the attribute. For other attributes, instances with missing values are removed under the constraint that at most 30% of the data may be removed in total. If too many instances would be removed, the attribute is removed instead. This procedure results in slightly modified datasets without missing values.

In case the algorithm can only process discrete attributes, continuous variables have to be discretized. This is done using the Weka-implementation of Fayyad & Irani's MDL method (Fayyad and Irani, 1992) as suggested in a benchmarking study on discretization by Liu et al (2002b). Algorithms that can handle both continuous and discrete attributes will sometimes process

[10] anneal, audiology, breast-cancer, cleveland-heart-disease, contact-lenses, credit, glass2, glass, hepatitis, horse-colic, hypothyroid, iris, krkp, labor, lymphography, monk1, monk2, monk3, mushroom, sick-euthyroid, soybean, tic-tac-toe, titanic, vote-1, vote, vowel, wine

[11] auto-mpg, autos, balance-scale, balloons, breast-w, breast-w-d, bridges2, credit-g, diabetes, echocardiogram, flag, hayes-roth, heart-h, heart-statlog, ionosphere, machine, primarytumor, promoters, segment, solar-flare, sonar, vehicle, zoo

[12] The UCI datasets are available at http://archive.ics.uci.edu/ml/. The Titanic dataset is available as part of the Delve project of the University of Toronto at http://www.cs.toronto.edu/~delve/.

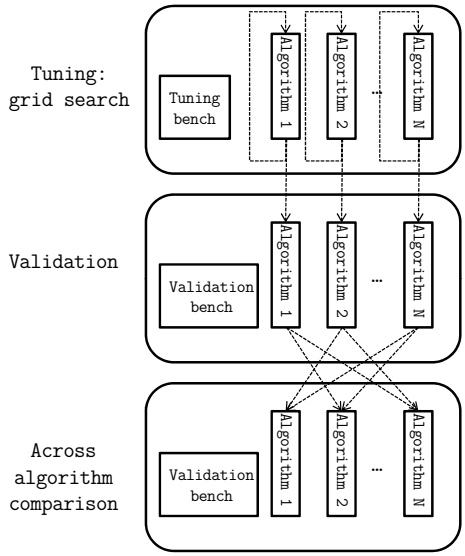


Fig. 1 The three stages of the experimental setup. Firstly, optimal default values for the parametrized heuristics are obtained using a tuning bench of datasets. Next the performance of these optimized heuristics is evaluated on an independent validation bench of datasets. Finally, we further test and compare these optimal heuristic values across all included algorithms on the same validation bench.

both types in different ways. Discretization is not used for these algorithms in order to capture all aspects of the algorithm.

The scale of the experiments is very large[13] as a result of the tuning of multiple parameters and the inclusion of multiple datasets. Because the execution time does not scale linearly in the number of attributes for most algorithms, feature selection is used. Feature selection is used in practice as it positively impacts the accuracy of the model (Witten and Frank, 2005). Martens et al (2007b) argue that for AntMiner+ and similar algorithms, datasets of 1000 instances should contain at most 20 attributes to obtain both reasonable accuracy and execution time. Using feature selection thus corresponds more closely to practical data mining situations. We opted for the Weka implementation of the reliefF attribute filter (Kira and Rendell, 1992; Kononenko, 1994) based on the results of a benchmarking study by Hall and Holmes (2003) and some preliminary experiments. As most of our datasets contain under 1000 instances, we use the Weka default and restrict the maximal number of input attributes to 10 for the more time-consuming algorithms. An overview of the preprocessing steps used for each algorithm is presented in Table 3.

[13] A number of 2 562 300 individual runs were executed in total.

Table 3 Preprocessing of the datasets for each algorithm

Algorithm	Discretization	Attribute selection
AntMiner+	✓	✓
AntMiner	✓	✓
PSO/ACO2		✓
HIDER		✓
RIPPER		

3.2 Search prodecure

Determining the optimal default parameter values is performed with a search procedure that optimizes a given objective function. For each selected parameter value, a ten times ten-fold cross-validation is run on each of the 27 tuning datasets. Janssen and Fürnkranz (2010) used a much faster single ten-fold cross-validation procedure. This change is necessary to counter the randomness introduced by the stochastic algorithms in this study. Furthermore, this approach has a small positive effect even for deterministic algorithms as the random shuffling of the data can produce very good/bad results by chance. This yields several possible measures per dataset, aggregated over the individual runs. We optimize for accuracy in the tuning phase. Measures to judge the comprehensibility of the rule set, such as the number of rules and average rule size, are balanced with accuracy as described in Section 3.5. The accuracy results for the 27 datasets then need to be aggregated into a single measure. This is done by taking the macro-average accuracy - the unweighted average - over the 27 datasets.

This objective function is maximized with a grid search procedure. Janssen and Fürnkranz (2010) assumed that the macro-average accuracy follows an inverse U-shape. Our assumptions is that this holds for the accuracy of individual datasets. The macro-average accuracy is however an aggregate of these individual inverse U-shapes and may have a more complex shape. The results in Section 4.1 indicate that this assumption holds for the aggregates of our datasets. The grid search works iteratively to find optimal heuristic parameters, as done for hyperparameters in SVMs in (Suykens et al, 2002) for example. In the first iteration, a given interval $[lower, upper]$ is scanned with a step size $step$. It then drills down in the region around the best result, $best$. The range for the second iteration is set to $[best - step, best + step]$ and the step size is decreased by a factor 5: $step = step/5$. This results in 8 new parameter values tested in the entire range between the two neighbors of the best found yet. This allows us to properly scan the entire interval, even if the curve is skewed - as is the case for the ACO-based algorithms and the relative cost measure. This is repeated until no further increase in macro-average accuracy is noted or after 5 iterations. This second stopping criterion is however never reached in our experiments, in part because the noise in the stochastic algorithms eventually surpasses smaller increases. For the Klösgen measure,

F-measure and relative cost measure, we search the range $[0; 1]$. For the m-estimate the range $[0; 10]$ is searched, although a wider range is used for the RIPPER algorithm after initial results showed that the optimal value was not in the range.

3.3 Comparing heuristics on multiple datasets

The resulting optimized heuristics are compared on the validation datasets to determine the best heuristic(s) for each algorithm. Because a new variant of the algorithm is created for each heuristic, this is a comparison of multiple algorithms on multiple datasets. Demšar (2006) makes a compelling case against t-tests with win/loss/tie characteristics and puts forward the non-parametric Friedman test as a more correct alternative. The Friedman test is based on the average ranks of the heuristics. All heuristics used perform equally well under the null-hypothesis and thus the average ranks should be equal. If the Friedman test indicates that the rankings are not equal, then it rejects the null-hypothesis. In this case at least two heuristics do not perform equally well.

Further post-hoc tests are needed to perform individual comparisons when the null-hypothesis is rejected. These tests focus on controlling the family-wise error introduced by performing multiple comparisons. In our case, we only compare each heuristic with the best-performing one, which results in 4 one-vs-one statistical tests. If the individual tests have a 5% chance of falsely rejecting the null-hypothesis, then the chance that at least one of these tests falsely rejects the null-hypothesis is about 18.5%. The p-values of the individual comparisons are adjusted for this family-wise error with the post-hoc Hommel test as implemented by García and Herrera (2008).[14]

3.4 Comparing parameter values across algorithms

Thirdly we perform a cross-algorithm comparison of the optimal default values, found using the procedure outlined in Section 3.2. To answer the question whether tuning is necessary for each algorithm individually, we propose a comparison between the tuning scenario and the non-tuning scenario. For each algorithm and parametrized heuristic, we perform further experiments on the validation datasets with the optimized values retrieved for the other algorithms. These represent different instances of the scenario in which a parameter value is taken directly from an existing algorithm, which is often the case as seen in the ACO-domain. We compare the results to those for the optimized heuristic value with the non-parametric Wilcoxon signed rank test. A large number of individual tests are performed which makes correcting for

[14] The statistical tool used to perform the Friedman test and advanced post-hoc procedures can be found at http://sci2s.ugr.es/keel/multipleTest.zip

the family-wise error without losing statistical power difficult. Because of this, the focus is on the general pattern and not on the individual comparisons.

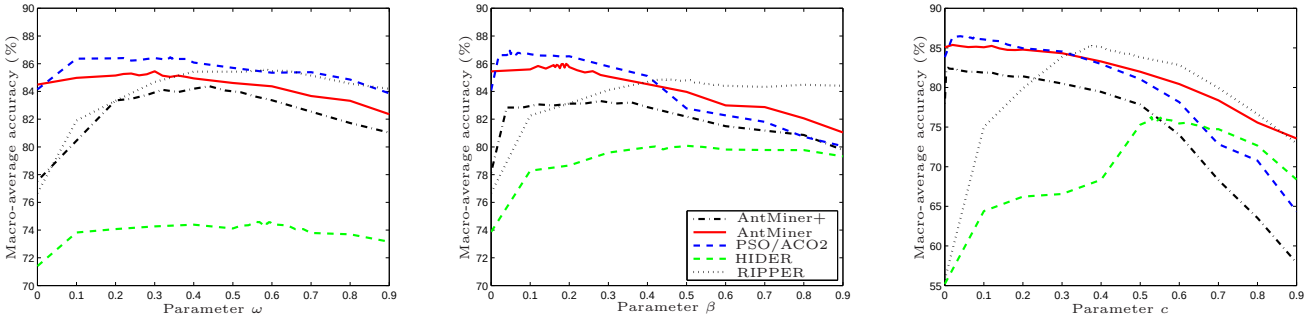
3.5 Comprehensibility vs. accuracy trade-off

The previous sections describe methods to determine parameter settings that are optimal for predictive accuracy. However rule models are often used because the comprehensibility of the model is important to the user as well, as observed by Baesens et al (2003a); Pazzani et al (2001); Verbeke et al (2011). In those cases, a model that sacrificies some accuracy for additional comprehensibility may be more suitable. This necessitates a trade-off between predictive accuracy and comprehensibility. This trade-off can be made by selecting a specific heuristic rule evaluation function. The parametrized heuristics discussed in this paper are especially suited to this task as the trade-off can be selected very precisely, which is not possible with most widely used heuristics.

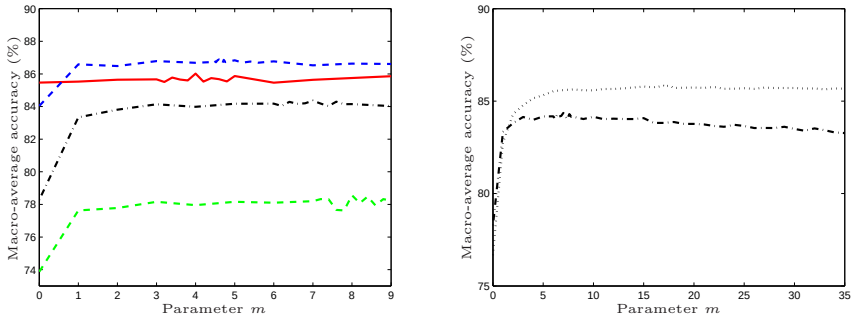
This trade-off is investigated by plotting a predictive accuracy measure against a comprehensibility measure for the tuning set results. For the accuracy measure, the macro-average accuracy is used. The comprehensibility is represented by the macro-average number of rules. Ranking measures are not used for two reasons. Firstly, performing this trade-off in practice requires an understanding of the size of the performance gap between competing settings. The ranking measures do not represent the size of the differences, while (macro-)averages do capture this. Secondly, the ranking results are influenced by the distribution of selected parameter values. Several competing measures can be used for comprehensibility and we have considered $\#R$, the number of rules, $\#T/R$ the average rule length and $\#T$ the total number of terms in the rule set. We will only show the macro-average number of rules as this is the most widely recognised measure and the other measures show similar behavior.

4 Results

This section presents the experimental results of the heuristic rule evaluation functions applied to the selected algorithms. For each algorithm and for each heuristic, an optimal parameter value was selected on a tuning bench. The results of this tuning procedure are shown in Section 4.1. The resulting parameter values are then tested on a validation bench in Section 4.2 in order to determine the best performing heuristic(s) per algorithm. Next, Section 4.3 contains an empirical comparison of the optimal parameter values across the different algorithms to determine the need for parameter tuning in other algorithms. Lastly, in Section 4.4 we analyse the trade-off between predictive accuracy and comprehensibility.



(a) Klösgen measure (b) F-measure (c) Relative cost measure



(d) m-estimate (e) m-estimate (RIPPER)

Fig. 2 Macro-average accuracy on the tuning bench for the different heuristics. The dash-dotted black line shows the results for AntMiner+. A solid red line is used for AntMiner. PSO/ACO2 and HIDER are represented by the blue and green (lower) dashed lines respectively. Ripper is shown in a dotted black line.

4.1 Parameter tuning

Optimal parameter values for each heuristic and each algorithm are obtained by the tuning procedure described earlier in Section 3. An overview of this tuning procedure is presented in Fig. 2. This figure shows the performance curves in terms of macro-average accuracy for each algorithm and each heuristic. Due to the random nature of the experimental measures and some degree of overfitting of these measures to the tuning bench, these curves only give an indication of the actual performance on new datasets. The true generalization behavior is given by the performance on the validation datasets in Section 4.2. The optimal parameter values are detailed in Table 4. The optimal parameter values for the ACO-based algorithms AntMiner, AntMiner+ and PSO/ACO2 are relatively close to each other and their performance curves are similar. On the other hand, HIDER and RIPPER seem to favor more coverage and thus larger parameter values. This (dis)similarity will be investigated further in Section 4.3.

As shown in Fig. 2a, the performance of the Klösgen measure increases starting from $\omega = 0$ until an algorithm-specific maximum is reached. For the algorithms included, these maxima are in the range $[0.3; 0.58]$. For higher values of ω, the performance decreases again. The results show that an optimal balance focuses more heavily on precision. Furthermore, Table 4 shows the ACO-based algorithms contained in this study benefit from even greater emphasis on precision than the other algorithms (lower ω).

Fig. 2b shows the performance curves of the F-measure. As with the Klösgen measure, the parameter β of the F-measure sets a trade-off between precision and coverage. The optimal parameter values for the algorithms included lie in the range $[0.048, 0.5]$. Thus the included algorithms benefit from emphasizing precision over coverage. The optimal parameter values for the ACO-based algorithms focus more on precision than those of HIDER and RIPPER. The optimum for the PSO/ACO2 algorithm is very close to $\beta = 0$ and focuses almost exclusively on precision.

The performance curves for the Relative cost measure are presented in Fig. 2c. A clear distinction can be made between the performance curves of the ACO-based algorithms and those of HIDER and RIPPER. In case of the former, the optimal values are in the range $[0.02, 0.06]$ and their performance curves decline steadily for larger values of c. These algorithms thus focus almost exclusively on the false positive rate. For the latter, a more balanced setting shows optimal performance at $c = 0.528$ and $c = 0.3728$ for HIDER and RIPPER respectively. Furthermore, these algorithms show bad performance for parameter values that focus more on the false positive rate.

The performance curves for the m-estimate are shown in Fig. 2d and Fig. 2e. This heuristic is an extension of the Laplace-corrected precision. For $m = 0$, the m-estimate equals precision and for $m = 2$ it closely resembles the Laplace-corrected precision. Higher values of m put more emphasis on coverage until the relative weighted accuracy is reached in the limit for $m \to \infty$ (Fürnkranz and Flach, 2005). The performance curves for AntMiner,

AntMiner+, PSO/ACO2 and HIDER in Fig. 2d are relatively flat in the range $[1, 10]$, with poor performance at $m = 0$ for most algorithms. The performance curve for RIPPER on the other hand shows a steady increase in this range and becomes flatter around the optimum at $m = 17.186$ before decreasing again. Because the performance curves for the m-estimate are relatively flat, the randomness introduced by the stochastic algorithms can have a large effect on the tuning procedure. The optimal values shown in Table 4 can thus exhibit a high degree of overfitting. Furthermore, the performance at $m = 2$ is relatively high for some algorithms and indicates that the Laplace-corrected precision could be a good choice for these algorithms.

In all graphs of Fig. 2, the performance curves for all algorithms except AntMiner show a poor performance for parameter values equal to zero, followed by a large increase for the next measurement. For a parameter value of zero, all heuristics equal either precision or the false positive rate. In this case, the heuristics ignore the number of examples covered and attempt to cover no false positives. For parameter values set to any $\delta > 0$ the heuristics will also take into account the number of examples covered. In the limit $\delta \to 0, \delta > 0$, the values returned by the heuristic rule evaluation functions will tend to precision (false positive rate). However, a bias towards rules with higher coverage will remain and act as a tiebreaker when several rules are found with the exact same precision (false positive rate). For example, the F-measure with $\beta = 0$ will not distinguish between multiple rules with the same precision. The selection of the rule then depends on the implementation details of the algorithm - most likely it will be either the first or the last rule found. A small positive β in turn will result in the F-measure selecting the rule with the highest coverage. This coverage bias steers the algorithm away from rules that cover only a few examples. These rules are most likely overfitting and thus the coverage bias decreases overfitting and increases performance.

The coverage bias has a small or no effect for AntMiner as observed by the absence of steep performance drops at zero in all graphs of Fig. 2. This is likely due to the $MinCases$ parameter used by AntMiner. The $MinCases$ parameter specifies the minimum number of examples covered by a rule which results in only rules with a sizeable coverage being evaluated. The effects of a coverage bias are less noticeable for these larger rules as ties are less common. The $MinCases$ parameter reduces overfitting by eliminating very small rules and thus performs a function similar to that of a coverage bias. PSO/ACO2 only implements a soft constraint through the $MinCases$ parameter and as such the coverage bias still has an effect as smaller rules are evaluated.

Table 4 Optimal parameter values for each algorithm and heuristic

Algorithm	Klösgen	F-measure	Relative cost	m-estimate
AntMiner+	0.44	0.28	0.028	7
AntMiner	0.3	0.192	0.02	4
PSO/ACO2	0.34	0.048	0.06	4.6
HIDER	0.58	0.5	0.528	8
RIPPER	0.576	0.448	0.3728	17.186

4.2 Results on the validation bench

In this section we evaluate the optimized heuristics on a validation bench of 23 UCI datasets. For each algorithm, we select the best-performing heuristic(s) out of the four optimized heuristics and the default heuristic. Each (algorithm,heuristic)-pair is evaluated on all 23 validation datasets using a ten times ten-fold cross-validation procedure. For each algorithm, the heuristics are then compared with a non-parametric Friedman rank test with post-hoc Hommel tests. In the next paragraphs, we first discuss these results per algorithm. The empirical results are presented in tables 5, 6, 7, 8 and 9. The best result per dataset in these tables is underlined and is compared to other results with a paired t-test. Results in bold/normal are not sigificantly different at the 5%/1% level while results in italic are sigificantly different at the 1% level.

Table 5 contains the results on all datasets for AntMiner+. The Klösgen measure appears to be the best performing heuristic on AntMiner+, both in terms of rank and macro-average accuracy. The m-estimate performs second-best and is close to the Klösgen measure. The F-measure does not perform significantly different, yet the difference in macro-average accuracy is 1.1%. The relative cost measure shows the worst performance for AntMiner+. *The Klösgen measure performs best, significantly improving the default heuristic with a 2.66% increase in macro-average accuracy (p-value 0.013). Using this measure, an improvement is obtained for 17.5 of the 23 datasets.*

Table 6 shows the results on all datasets for AntMiner. Based on the rank test, the m-estimate shows the best performance in AntMiner. However, both the Klösgen measure and the F-measure report a slightly higher macro-average accuracy and the relative cost measure only performs slightly worse. As none of these differences are significant, these heuristics perform equally well for AntMiner. *The m-estimate performs best, significantly improving the default heuristic with a 3.19% increase in macro-average accuracy (p-value 0.001). Using this measure, an improvement is obtained for 19 of the 23 datasets.*

Table 7 presents the results on all datasets for PSO/ACO2. The Friedman test finds no significant differences in performance between the heuristics. All reported differences are too small to be meaningful. *The Klösgen measure performs best in terms of macro-average accuracy with a 0.18% increase in macro-average accuracy (p-value 0.93), though the default Laplace heuristic performs best in terms of rank. As observed by the statistical tests, the observed differences are not significant. Using the Klösgen measure, an improvement is obtained for 11.5 of the 23 datasets.*

Table 8 shows the results on all datasets for HIDER. The best performing heuristic for HIDER is the F-measure, both in terms of rank and in terms of macro-average accuracy. The F-measure significantly outperforms all other optimized heuristics. The 0.65% difference in macro-average accuracy with the default heuristic is however not significant at the 10% level with a p-value of 0.16. *The F-measure performs best, improving the default heuristic*

Table 5 Experimental accuracy results for AntMiner+

Dataset	Klösgen	m-estimate	F-measure	AntMiner+	Relative cost
auto-mpg	**70.88**	**70.64**	69.60	*66.95*	*66.36*
autos	80.27	80.61	*80.07*	**82.41**	78.97
bal	79.44	79.75	**80.63**	*78.46*	**80.70**
balloon	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
bcw	**95.46**	**95.58**	**95.52**	**95.40**	**95.32**
bridges2	**52.25**	**51.50**	*49.25*	**51.75**	50.13
echocardiogram	**70.64**	**71.92**	**70.34**	*69.25*	**70.45**
flag	**64.18**	**63.59**	**63.35**	*60.51*	*61.30*
ger	**72.16**	71.05	71.53	*70.80*	*70.50*
hayes-roth	**85.88**	*84.69*	*81.69*	*80.00*	*81.38*
heart-stat	**78.59**	**78.33**	**77.74**	**78.48**	77.15
heart-h	**76.27**	**76.40**	**76.44**	**76.25**	**76.25**
ionosphere	**88.83**	**88.09**	**88.98**	**88.94**	**88.92**
machine	38.50	**40.08**	38.74	**39.52**	*35.59*
pima	**72.85**	*69.70*	**72.38**	*68.62*	*69.48*
primary-tumor	**34.96**	**35.59**	*23.37*	*25.22*	*20.74*
promoters	**87.67**	85.87	**87.16**	**87.66**	84.94
segment	86.29	**87.79**	**85.77**	85.69	*72.94*
solar-flare	**72.77**	*70.14*	*69.94*	*52.95*	*49.54*
sonar	70.44	**71.32**	**71.78**	**71.40**	**71.83**
vehicle	**64.13**	61.81	62.88	*51.16*	*51.43*
wdbc	**93.31**	92.83	**93.13**	**93.34**	*92.86*
zoo	**95.32**	*93.63*	**95.23**	**94.94**	**95.45**

Table 6 Experimental accuracy results for AntMiner

Dataset	m-estimate	F-measure	Klösgen	Relative cost	Sens×spec
auto-mpg	**75.57**	**75.93**	*74.91*	*73.42*	**75.17**
autos	**74.41**	**74.38**	**74.75**	**74.48**	*66.25*
bal	**82.96**	*79.30*	*78.78*	*79.50*	*68.52*
balloon	**81.50**	**82.00**	**85.00**	**83.50**	**78.50**
bcw	**95.45**	**95.18**	**95.10**	**95.32**	*91.51*
bridges2	**59.25**	**58.63**	**58.88**	**58.88**	**59.38**
echocardiogram	**76.26**	**76.63**	**76.23**	**75.90**	**76.55**
flag	**63.90**	**62.01**	**63.58**	*60.88*	*60.31*
ger	*70.55*	**72.60**	**71.91**	*69.26*	*70.41*
hayes-roth	*77.06*	**80.94**	**81.88**	**80.75**	*67.94*
heart-stat	**78.85**	**79.22**	**79.07**	**78.19**	**77.15**
heart-h	**78.04**	**77.23**	**77.95**	**77.73**	**77.70**
ionosphere	**89.96**	**90.69**	*88.95*	**90.76**	*87.86*
machine	**38.77**	*37.04*	**38.39**	36.85	**37.41**
pima	**76.09**	**75.83**	**76.31**	**76.09**	75.14
primary-tumor	**36.04**	**36.48**	**36.00**	**35.67**	34.78
promoters	**85.56**	**86.25**	**85.25**	**84.83**	*79.69*
segment	**89.46**	89.08	*87.24*	88.94	*80.26*
solar-flare	*72.48*	*72.82*	*71.26*	*71.99*	**74.25**
sonar	*76.09*	**77.05**	**78.07**	76.58	*75.06*
vehicle	**66.06**	**66.35**	**65.99**	**66.25**	*57.60*
wdbc	**94.50**	**94.41**	93.96	**94.46**	*93.39*
zoo	90.21	**90.35**	**91.14**	**91.44**	**90.86**

Table 7 Experimental accuracy results for PSO/ACO2

Dataset	Laplace	Klösgen	F-measure	m-estimate	Relative cost
auto-mpg	**80.26**	*78.27*	79.08	78.61	**79.62**
autos	**82.33**	**81.57**	**81.78**	**81.03**	**82.72**
bal	**79.41**	*77.41*	**80.03**	*79.35*	**80.13**
balloon	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
bcw	**94.83**	**94.48**	**94.92**	**94.47**	*94.07*
bridges2	*60.75*	**63.63**	**63.00**	*60.88*	**62.63**
echocardiogram	**76.86**	**76.52**	**75.28**	**76.63**	*73.77*
flag	*60.83*	**63.09**	61.38	**64.46**	62.50
ger	**72.66**	*71.63*	**72.10**	**72.29**	71.36
hayes-roth	66.50	**67.63**	65.94	*65.25*	**66.81**
heart-stat	**80.56**	**80.78**	**80.22**	**79.93**	**79.59**
heart-h	**78.28**	**79.11**	**78.46**	77.88	*75.88*
ionosphere	**91.82**	**92.42**	**91.77**	**92.13**	91.68
machine	**19.54**	**19.40**	19.25	**19.64**	**19.11**
pima	*73.22*	**74.52**	*72.98*	*73.12*	*72.16*
primary-tumor	**36.63**	**36.52**	**37.15**	**36.71**	**36.30**
promoters	**83.96**	**84.93**	**84.26**	*82.39*	**82.95**
segment	**96.39**	*95.38*	**96.35**	**96.39**	*95.75*
solar-flare	*73.73*	**74.40**	*73.59*	73.87	*72.92*
sonar	**75.95**	**77.60**	**76.96**	**77.15**	**75.78**
vehicle	**69.61**	**69.17**	**69.19**	68.73	**69.83**
wdbc	95.01	**95.33**	94.92	94.59	**95.68**
zoo	**90.81**	**90.38**	**90.41**	**90.60**	**90.90**

Table 8 Experimental accuracy results for HIDER

Dataset	F-measure	HIDER	m-estimate	Relative cost	Klösgen
auto-mpg	**74.02**	**73.10**	72.79	*71.26*	*72.62*
autos	**76.21**	*73.95*	74.76	*68.27*	**75.33**
bal	**71.62**	*68.47*	*69.00*	**70.90**	*68.98*
balloon	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
bcw	*93.40*	**95.32**	*91.98*	*92.03*	*90.46*
bridges2	**60.63**	**59.38**	**59.50**	*52.75*	**59.25**
echocardiogram	67.34	**70.50**	*62.51*	67.03	*60.16*
flag	**53.60**	**54.00**	**52.74**	*48.62*	**52.47**
ger	*70.89*	**71.20**	*70.10*	*69.79*	*64.34*
hayes-roth	**80.28**	*75.28*	*77.47*	*75.52*	*77.28*
heart-stat	**74.58**	**74.80**	*72.32*	**73.99**	*71.21*
heart-h	**72.15**	**72.61**	*66.13*	**71.91**	*68.19*
ionosphere	**78.49**	77.04	*69.97*	*75.55*	*64.16*
machine	**36.67**	**35.93**	*33.82*	**36.77**	**35.71**
pima	**71.16**	**70.92**	*65.84*	**69.97**	*68.01*
primary-tumor	*33.82*	*30.56*	**34.90**	**34.23**	**35.52**
promoters	**79.77**	76.16	*75.00*	**78.64**	*67.61*
segment	*89.67*	*87.36*	**91.05**	*81.62*	*87.66*
solar-flare	**73.16**	72.83	*71.45*	*64.39*	*70.92*
sonar	*55.11*	**58.54**	*50.72*	*52.11*	*44.29*
vehicle	**65.28**	64.43	63.84	*59.55*	64.61
wdbc	**89.65**	**89.83**	**89.40**	89.24	*86.75*
zoo	89.06	**89.33**	*87.58*	**90.19**	**89.18**

Table 9 Experimental accuracy results for RIPPER

Dataset	Information gain	Klösgen	m-estimate	F-measure	Relative cost
auto-mpg	**78.67**	*76.25*	*77.04*	*76.51*	*74.18*
autos	**76.16**	**76.10**	**75.16**	*71.19*	*72.83*
bal	**72.64**	*71.22*	72.03	71.90	*69.10*
balloon	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
bcw	*93.94*	94.89	*94.35*	**95.46**	*93.16*
bridges2	58.38	**59.88**	60.00	**59.50**	**60.63**
echocardiogram	**77.76**	**77.85**	**77.57**	**76.07**	**77.38**
flag	*61.24*	*60.62*	*60.10*	**63.20**	*59.48*
ger	71.54	**72.41**	**71.96**	71.98	*71.18*
hayes-roth	**83.13**	*80.19*	81.38	*75.44*	*79.06*
heart-stat	**75.59**	**75.07**	**74.48**	**75.19**	**75.15**
heart-h	*76.44*	*76.25*	**78.43**	**78.16**	*75.21*
ionosphere	*89.15*	*89.57*	*90.37*	**91.48**	*90.57*
machine	**30.38**	**30.00**	**29.95**	**30.43**	**29.57**
pima	**74.74**	**74.67**	*73.50*	**74.31**	**74.23**
primary-tumor	**35.00**	**34.63**	**34.26**	*33.52*	*33.37*
promoters	**82.64**	**82.92**	83.21	82.17	**84.62**
segment	**95.23**	94.97	*94.03*	*93.78*	*94.34*
solar-flare	*69.88*	**70.99**	*69.78*	**71.10**	*69.98*
sonar	**73.85**	**74.76**	**74.86**	*71.59*	**74.66**
vehicle	**68.42**	**68.35**	**68.72**	**68.16**	*67.38*
wdbc	*94.24*	*93.67*	*94.22*	**95.08**	*93.74*
zoo	**89.60**	**88.32**	**89.21**	87.62	87.72

with a 0.65% increase in macro-average accuracy (p-value 0.16). Using the F-measure, an improvement is obtained for 13.5 of the 23 datasets.

Table 9 contains the results on all datasets for RIPPER. The standard information gain heuristic of RIPPER performs best, both in terms of rank and in terms of macro-average accuracy. The Klösgen measure, m-estimate and F-measure show a slightly lower macro-average accuracy and do not perform significantly worse. The relative cost measure on the other hand performs significantly worse. *The default information gain metric performs best in terms of rank and macro-average accuracy, with a 0.18% and 0.22% increase over the m-estimate and Klösgen measure respectively (both p-value 0.35). Using the Klösgen measure, an improvement is obtained only for 8.5 of the 23 datasets.*

The largest improvement in performance was obtained for AntMiner that uses the $sensitivity \times specificity$ heuristic as a default. Holden and Freitas (2008) report selecting the Laplace heuristic for PSO/ACO2 because of low performance with $sensitivity \times specificity$. We compare the $sensitivity \times specificity$ heuristic with the default and best performing heuristic for the ACO-based algorithms. Each of these algorithms is evaluated on the validation bench. Results are compared with a Wilcoxon signed-rank test at the 5% level. Table 10 presents the results of this comparison. Significant differences are shown in bold and underlined. $Sensitivity \times specificity$ is always significantly outperformed and the differences in macro-average accuracy are even larger for AntMiner+ and PSO/ACO2. These results show the general poor performance of the $sensitivity \times specificity$ heuristic in ACO-based algorithms.

Table 10 Performance of sensitivity×specificity

Algorithm	Sensitivity×specificity	Default	Best
AntMiner+	65.94	**72.60**	**75.26**
AntMiner	71.99	71.99	**75.18**
PSO/ACO2	69.62	**75.65**	**75.65**

The overall results are presented in Table 11, which contains the four parametrized heuristics with optimal values and the original heuristic for each algorithm listed as 'Default'. In this table, the top-performing heuristic is underlined. All heuristics are compared to this heuristic and the p-values are adjusted to correct the family-wise error. Results in bold/normal font are not significantly different from the best results at the 10%/1% level. All other results, noted in italic font, are significantly different from the best at the 1% level. These results show three candidate heuristics for implementation in new algorithms: F-measure, Klösgen measure and m-estimate. We do not recommend using m-estimate due to possible complications with parameter setting as discussed in Section 2. The F-measure performs better than the Klösgen measure in HIDER and is never significantly worse than the best heuristic. The Klösgen measure on the other hand might perform slightly better on the ACO-based algorithms. Based on the results on the validation bench, both heuristics are a good choice.

4.3 Comparing parameter values

In this section, we empirically compare the optimal parameter values as found in Table 4. For each heuristic and for each algorithm, the performance is measured when using the optimal values for the four other algorithms. This represents a scenario in which no tuning was done for an algorithm, but rather an existing value was imported from another algorithm. These results are then compared with those using optimal settings for the algorithm in order to determine the impact of performing tuning on the algorithm itself as opposed to importing values from related algorithms.

Tables 12, 13, 14 and 15 show the results of this comparison for the F-measure, Klösgen measure, m-estimate and relative cost measure respectively. In these tables, each row represents a single algorithm and each column a single parameter value, with the cell showing the measurement for the combination of both. Both rows and columns are ordered by ascending parameter values. The

Table 11 Comparing heuristics on the validation bench for the optimal parameter values

	Default		F-measure		Klösgen		m-estimate		Relative cost	
Algorithm	Accuracy	Rank	Accuracy	Rank	Accuracy	Rank	Accuracy	Rank	Accuracy	Rank
AntMiner+	*72.60*	*3.46*	**74.15**	**2.83**	**75.26**	**2.13**	**74.82**	**2.65**	71.40	3.93
AntMiner	71.99	4.17	**75.23**	**2.52**	**75.24**	**2.76**	**75.18**	**2.39**	**74.85**	**3.15**
PSO/ACO2	**75.65**	**2.61**	**75.61**	**3.04**	**75.83**	**2.65**	**75.48**	**3.17**	**75.31**	**3.52**
HIDER	**71.37**	**2.43**	**72.02**	**1.78**	68.47	3.87	69.69	3.43	69.32	3.48
RIPPER	**75.16**	**2.39**	**74.51**	**3.09**	**74.94**	**2.83**	**74.98**	**2.87**	74.24	3.83

cells contain $MA - MA_{opt}$, the difference in macro-average accuracy between the specified parameter value and the optimal parameter value. The diagonal is crossed out as in this case both parameter values are equal. For each cell, we statistically compare the results with the non-parametric Wilcoxon signed rank test. Differences significant at the 10% level are highlighted in boldface.

Table 12 shows that the performance of PSO/ACO2 and AntMiner drops for the higher parameter values found for RIPPER and HIDER. An equal drop in performance is seen for HIDER at lower parameter values. The macro-average accuracy for RIPPER drops 2.3% for $\beta = 0.048$, but with a p-value of 0.13 this is not significant. The lower macro-average accuracy is however consistent with the results on the tuning bench as illustrated by Fig. 2b. Inspection of the results learns that several datasets are sensitive to lower parameter values which results in a large difference in macro-average accuracy.[15] The Wilcoxon test is less sensitive to these outliers and focuses more on the other datasets. Using RIPPER with this parameter setting works well for many problems, but with the risk of large performance decreases for a few. Consistent with the tuning bench results, the gap in parameter values from $\beta = 0.048$ to $\beta = 0.192$ between PSO/ACO2 and AntMiner does not further distinguish the ACO-based algorithms.

As for the F-measure, the results for the Klösgen measure in Table 13 again show a significant decrease in performance at lower parameter values for HIDER. The performance of RIPPER is unaffected by ω in the range $[0.3; 0.58]$. The parameter values are interchangeable for the ACO-based algorithms. The results for AntMiner are somewhat inconclusive as for nearly similar parameter values 0.576 and 0.58, both a significant (p=0.055) and non-significant (p=0.39) performance drop are found. Fig. 2a shows a small performance drop for AntMiner at these values, corroborating that the difference is not an outlier.

[15] The datasets *autos*, *flag*, *sonar* and *vehicle* show differences exceeding 10%.

Table 12 Comparing optimal settings for the F-measure across algorithms

	0.048	0.192	0.28	0.448	0.5
PSO/ACO2		0.03	-0.03	-0.87	**-1.56**
AntMiner	-0.23		-0.09	**-1.13**	-0.82
AntMiner+	-0.14	0.30		0.31	0.35
RIPPER	-2.30	-0.58	0.25		0.17
HIDER	**-1.80**	**-0.95**	**-0.83**	**-0.23**	

Table 13 Comparing optimal settings for the Klösgen measure across algorithms

	0.3	0.34	0.44	0.576	0.58
AntMiner		-0.33	0.07	**-0.56**	-0.33
PSO/ACO2	-0.10		-0.10	-0.35	-0.36
AntMiner+	-0.50	-0.44		-0.74	-0.54
RIPPER	0.18	0.19	0.07		0.07
HIDER	**-1.52**	**-1.27**	**-0.77**	-0.08	

Table 14 Comparing optimal settings for the m-estimate across algorithms

	4	4.6	7	8	17.186
AntMiner		-0.13	-0.47	-0.31	-0.19
PSO/ACO2	0.03		0.18	0.05	-0.23
AntMiner+	0.09	-0.08		0.15	-0.40
HIDER	-0.18	-0.01	0.06		**0.44**
RIPPER	-0.10	-0.11	-0.2	0.08	

Table 15 Comparing optimal settings for the relative cost measure across algorithms

	0.02	0.028	0.06	0.3728	0.528
AntMiner		-0.20	-0.22	-0.93	**-2.58**
AntMiner+	0.31		**-1.22**	**-6.22**	**-10.5**
PSO/ACO2	0.08	0.04		**-3.33**	**-5.99**
RIPPER	**-14.27**	**-13.24**	**-10.97**		**-2.11**
HIDER	**-7.49**	**-6.95**	**-6.50**	**2.47**	

Table 14 presents the results for the m-estimate. Both the macro-average accuracy and the Wilcoxon tests show that the found parameter values are largely interchangeable across all algorithms in this study. This is consistent with the flat performance on the tuning bench in Fig. 2d. It should be noted that the performance of RIPPER drops around $m = 5$ on the tuning bench in Fig. 2e, so the results for $m = 4$ are probably borderline. The only significant difference is observed for HIDER at higher parameter values with a relatively small 0.44 increase (p=0.077).

Unlike those for the m-estimate, the results for the relative cost measure in Table 15 are significant. Consistent with the results on the tuning bench in Fig. 2c, a clear distinction can be made between the ACO-based algorithms, and HIDER and RIPPER, respectively. The former favor very low parameter values that focus almost exclusively on the false negative rate while the latter prefer larger, more balanced parameter settings. The parameter difference between RIPPER and HIDER is significant as well. The ACO-based algorithms favor parameter values in the small range $[0.02; 0.06]$. A notable result is the difference in performance for AntMiner+ for $c = 0.06$, which is significant according to the Wilcoxon test (p=0.031). The difference in parameter values is however seemingly too small to justify this result. Furthermore, this performance drop is absent in the tuning bench results. This further supports our claim that this result is likely an outlier.

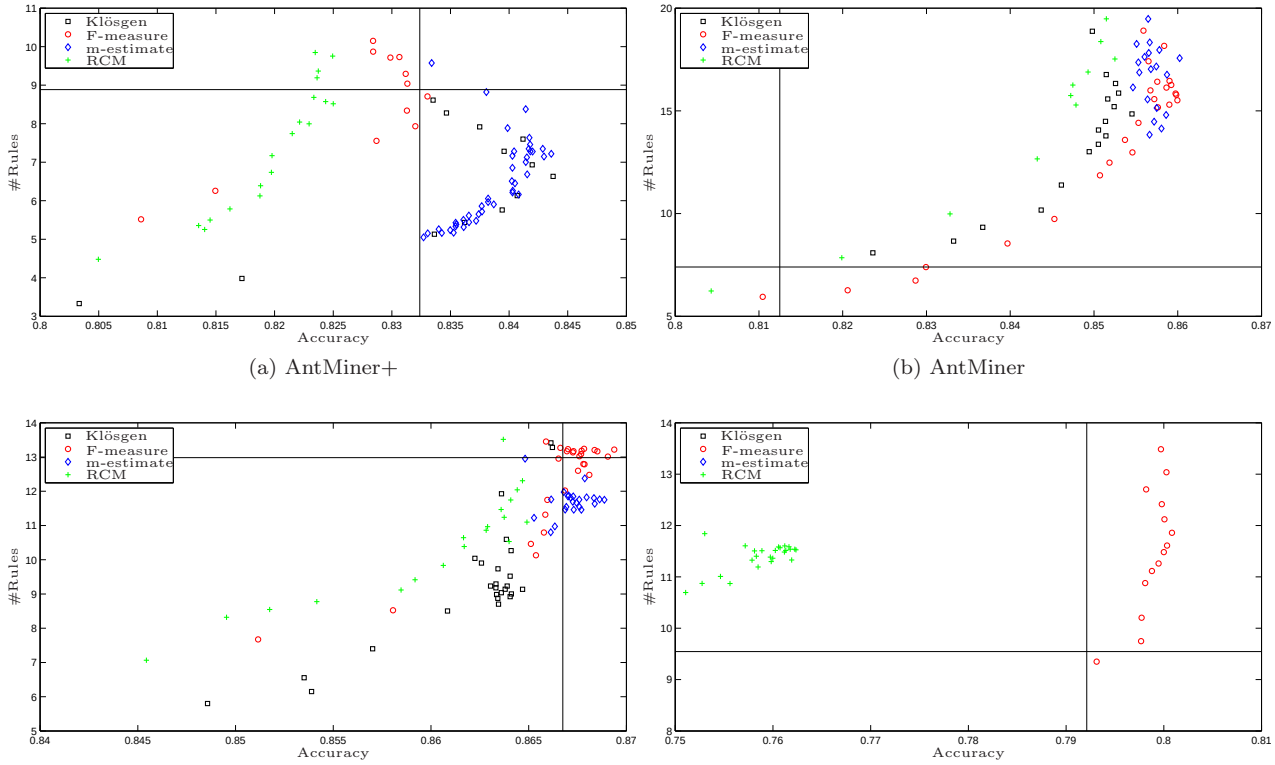
Overall, we observe that the ACO-based algorithms PSO/ACO2, AntMiner and AntMiner+ favor parameter values that focus more on confidence, while RIPPER and HIDER put more emphasis on coverage. The results in this section show that using values optimized for the former into the latter or vice versa, leads to a significant reduction in performance. Furthermore, we find no such difference in performance when interchanging parameters between the ACO-based algorithms.

4.4 Accuracy vs. comprehensibility trade-off

The comprehensibility vs. predictive accuracy trade-off is a multi-criteria decision making problem. We are interested in the nondominated settings, for which improving one criterion is not possible without sacrificing the other criterion (Steuer, 1986). We determine the Pareto front of nondominated solutions for the four metaheuristic-based algorithms based on Fig. 3. Each point in these graphs shows the macro-average accuracy and the macro-average number of rules on the tuning set for one setting. Not all settings are shown as the ranges for the axes are selected to show the region that is of interest and thus differs per algorithm. For each algorithm, the performance of the original algorithms is represented by the horizontal and vertical line. These lines divide the graphs in four quadrants located northwest (NW), northeast (NE), southwest (SW) and southeast (SE) relative to the original algorithm. The SE quadrant is the most interesting as it contains the cases in which the new setting dominates the original setting. For these settings, both the predictive accuracy and the comprehensibility are better than for the original. Likewise, the settings in the NW quadrant are dominated by the original setting and should not be considered. The settings in the SW and NE quadrant perform better on one measure and worse on the other. Thus a choice between both involves a trade-off. The nondominated settings are located in the SW, SE and NE quadrants and represent an entire array of settings that range from high comprehensibility and low accuracy to low comprehensibility and high accuracy. The use of parametrized heuristics allows us to reach intermediary points between any two settings and as such these points are representatives of a Pareto curve.

Fig. 3a shows the trade-off for AntMiner+. As already observed in section 4.2, the parametrized heuristics improve the predictive performance of the algorithm. The most promising solutions are the Klösgen measure with $\omega \in [0.44, 0.6]$ and m-estimate with $m \in [7, 35]$. For larger parameter settings, even more comprehensibility is found, but at a higher accuracy cost. For smaller parameter values, both the accuracy and the comprehensibility deteriorate due to overfitting. Thus both the predictive accuracy and the comprehensibility are improved for AntMiner+.

Fig. 3b shows the trade-off for AntMiner. The improvement in predictive accuracy is again observed - consistent with the results in section 4.2. This time, the most promising solutions are the m-estimate with $m \in [4, 10]$ and the F-measure with $\beta \in [0.192, 0.9]$. Higher parameter values can be used for further comprehensibility. For the m-estimate in particular, not the entire usefull range is sampled by the tuning experiments. We observe that the original heuristic focuses heavily on comprehensibility. However this trade-off is suboptimal as it is dominated by the F-measure in the range $\beta \in [0.6, 0.8]$. For $\beta = 0.6$ a 1.75% higher accuracy can be obtained for roughly the same comprehensibility. Alternatively $\beta = 0.8$ offers a gain in comprehensibility of 1.13 rules with a 0.81% higher accuracy. Again, both the predictive accuracy and the comprehensibility are improved for AntMiner.



(a) AntMiner+ (b) AntMiner

(c) PSO/ACO2 (d) HIDER

Fig. 3 Macro-average accuracy vs. macro-average number of rules trade-off for the metaheuristic-based algorithms. The performance of the original algorithm is represented by the horizontal and vertical line.

Fig. 3c shows the trade-off for PSO/ACO2. While several settings are situated in the SE quadrant, the observed accuracy differences are very small, consistent with the results in section 4.2. In terms of accuracy, no real improvement can be found. A first cluster of promising solutions is formed by the m-estimate settings. This cluster dominates a cluster of F-measure solutions and the original heuristic due to a comprehenisibility gain of 1-2 rules. A second promising cluster is formed by the solutions of the Klösgen measure as a comprehensibility gain of 4-5 rules is found for a small 0.1% accuracy loss. For higher parameter settings that focus less on accuracy, the Klösgen measure provides the most promising settings. As the original Laplace heuristic has a high focus on accuracy, we find that we can greatly improve the comprehensibility of PSO/ACO2 at no or a small accuracy cost.

Fig. 3d shows the trade-off for HIDER. As seen in section 4.2, only the F-measure performs reasonable in this algorithm. We find that the original heuristic already focuses more on comprehensibility. The most promising solutions are in the range $\beta \in [0.5, 0.9]$ for which a higher accuracy is found. On the validation set, we observed a borderline non-significant improvement with p-value 0.16. However, we observe that on the tuning set, a similar large enough improvement is observed for all points in the range $\beta \in [0.1, 0.9]$. This suggests that the performance difference may be a real difference and the non-significant result was due to the lower power of the test. The number of rules $\#R$ of F-measure with $\beta = 0.9$ shows a small improvement of the original heuristic, but the average rule length $\#T/R$ is 0.71 terms shorter for the original heuristic. As such the comprehensibility of the original heuristic is not improved upon. For higher parameter settings, even more comprehensibility is likely to be found. The parametrized heuristics do not improve the comprehensibility for HIDER, but higher accuracy settings can be reached at some comprehensibility cost.

5 Discussion

5.1 Impact on existing sequential covering algorithms

Our results have some interesting implications for current sequential covering algorithms. Most notably, the importance of a good heuristic rule evaluation function is demonstrated, which is largely left unconsidered in existing research. More specifically, we argue against the (further) use of the sensitivity times specificity heuristic, indicate that the Laplace metric performs well for specific algorithms only, and propose fine-tuning of parametrized heuristics.

Most ACO-based algorithms use the multiplication of sensitivity and specificity as proposed in the first ACO-based rule induction technique, AntMiner. Our results show this heuristic is not very suitable for accurate modelling and can be significantly improved for the AntMiner and AntMiner+ techniques. When accuracy is of lesser importance, other settings are still superior in terms of both comprehensibility and accuracy. This is consistent with the results of

an experiment with several non-parametrized heuristics on ACO-based algorithms (Salama and Abdelbar, 2011). Similar improvements are likely to be observed for other techniques using the same heuristic, such as AntMiner2 (Liu et al, 2002a), AntMiner3 (Liu et al, 2003), cAntMiner (Otero et al, 2009) and PSO-Sousa (Sousa et al, 2004). This heuristic should no longer be used in any (new) ACO-based rule induction technique.

The Laplace metric performs well, as can be seen from the PSO/ACO2 results. However, the tuning bench results for the F-measure suggest this performance does not generalise to other algorithms. For smaller values of β, both heuristics are nearly equivalent. We observe that both HIDER and RIPPER only perform well for higher values of this parameter. This is supported by the low performance of the Laplace metric on the CN2 algorithm reported by Janssen and Fürnkranz (2010). Furthermore we find that the m-estimate and the Klösgen measure offer a relatively large increase in comprehensibility at no or a minor accuracy cost. As the Laplace metric is closely related to both the F-measure and m-estimate for very low parameter settings, it is geared towards the extraction of many smaller rules, which negatively impacts the comprehensibility. For this reason, we propose a less extreme parameter setting for PSO/ACO2.

Our results on the validation datasets show that for each algorithm no single heuristic performs best on all datasets. When computational time and power is available, tuning across heuristics and parameter values is therefore surely a very valid methodological preprocessing step. As such, we suggest to include an option for the user to tune the heuristic rule evaluation function. In some cases, we can approximate these optimal tuning results with a generic heuristic and value, as discussed next.

5.2 Suggested generic rule evaluation heuristic and value

Although no best heuristic rule evaluation function can be provided for all algorithms, within a category of ACO-based algorithms this is possible. For the ACO-based algorithms, we observe that the Klösgen measure, F-measure and m-estimate with adequate parameter setting provide the best macro-average accuracy results on the validation datasets, with no significant differences between them. We prefer the Klösgen measure for several reasons. For the ACO-based algorithms, the Klösgen measure has the highest average rank - being the best of three on AntMiner+ and PSO/ACO2 - and shows general high macro-average accuracy for all tested algorithms in this class, including AntMiner. Furthermore, the m-estimate and F-measure might generate problems with parameter setting for larger datasets. The optimal Klösgen parameter values for AntMiner, PSO/ACO2 and AntMiner+ do not yield significantly different results and any of these values could be chosen. Based on the shape of the macro-average accuracy over the range of values, we decide to choose the median value of the three tuned values, which is 0.34. In conclusion, for the

ACO-based algorithms, we propose to use the Klösgen measure with parameter value ω equal to 0.34 when predictive accuracy is the primary concern.

Our results show that genetic algorithms and construction algorithms both require heuristics that assign more weight to coverage. HIDER also seems to be more sensitive to both the choice of heuristic and the parameter value, which can be explained by the absence of pruning. We suggest the F-measure with parameter value β set to 0.5 for the genetic algorithms. However this is based only on the results for the HIDER algorithm. As this may not generalize to other genetic algorithms, it is advisable to perform parameter tuning in the suggested region.

The construction algorithms are a very diverse class, so no general heuristic can be suggested based on the results for RIPPER alone - if at all. The results of Janssen and Fürnkranz (2010) on several CN2 implementations indicate good performance for the m-estimate and Klösgen measure. We suggest parameter values for ω and β in the ranges $[0.4323; 0.576]$ and $[0.448; 0.5]$ respectively. However, the results for RIPPER show generally high performance for the information gain. This gain heuristic successfully leverages upon the extra information provided by the previous rule to achieve high performance. We advise to test gain heuristics whenever they are applicable in an algorithm.

5.3 Accuracy vs. comprehensibility

The predictive accuracy is in many cases not the only concern of the user as good model comprehensibility is often desired. As such we discuss the merits of parametrized heuristics in finding a good balance. Firstly our selection of parametrized heuristics enables us to improve the balanced solutions in terms of accuracy and/or comprehensibility on the the algorithms involved. Our results clearly show that for AntMiner+ and AntMiner, we both improve the accuracy and comprehensibility with these heuristics. While the accuracy of PSO/ACO2 is only matched and not improved, we offer a much higher comprehensibility. Likewise, we find solutions with higher accuracy for HIDER, though at a comprehensiblity cost. Secondly, these parametrized heuristics offer an unmatched flexibility when it comes to setting the trade-off between accuracy and comprehensibility. As in (Salama and Abdelbar, 2011), we can identify those heuristics that are on the Pareto front. Of the four heuristics on the Pareto front in their experiment, three are equivalent to specific parameter settings that are dominated in our experiments. The parametrized heuristics additionally offer a continuous trade-off where other heuristics only offer a choice between a limited set of fixed trade-offs.

Whether it is up to the algorithm designer or the end user to decide on this trade-off is still an open question. From a usability perspective, it is advisable to at least provide a good default setting that satisfies the designer's view on this trade-off. In addition, we advocate to allow an expert user to optimize this parameter for a given task. This offers the previously discussed performance benefits of tuning on individual datasets and also allows the user

to select his own optimal balance of accuracy and comprehensibility, using the parameter of the heuristic as a slider setting. Finally, our results show that the balance between accuracy and comprehensibility has a severe impact on an algorithm's performance with respect to these metrics. Where fixed heuristics limit an algorithm to tasks that require a specific mix of comprehensibility and accuracy, parametrized heuristics allow each algorithm to be used in a wider range of tasks. All in all, the heuristic parameter thus performs a role similar to C4.5's pruning confidence factor and adds a greater flexibility to rule learning algorithms if it can be user defined. Whether it is the end user or the algorithm designer, our data[16] allow someone to quickly select either a specific setting or a region of interest.

6 Conclusion

Over the last decades, many sequential covering algorithms have been proposed. Only few researchers have recognized the importance of a proper rule evaluation function and its implications for the overall performance. Although no silver bullet heuristic exists that shows optimal accuracy and comprehensibility for all algorithms, we find heuristics that improve upon the accuracy and/or comprehensibility for the metaheuristic-based algorithms. A key result is that the algorithms that use ant colony optimization are similar enough with respect to rule evaluation to share a good default heuristic for this class. This finding alleviates the problem of selecting a heuristic for a new algorithm as we can identify a good starting heuristic.

The multiplication of sensitivity and specificity, often used in ACO-based algorithms, is shown to be unsuitable for this task. For this category of rule induction techniques, we propose to use the Klösgen measure with parameter value 0.34, which significantly improves existing ACO-based algorithms AntMiner and AntMiner+. For genetic algorithms, we advise the F-measure with parameter value 0.5 (and additional tuning in this region), which shows a (non-significant) 0.65% difference for HIDER.

A significant contribution is made to the problem of setting a trade-off between comprehensibility and accuracy. Parametrized heuristics offer a previously unseen flexibility to the algorithm designer or end user to select this trade-off and as a result Pareto dominate many fixed heuristics. The data from our experiments allows a quick trade-off between both metrics for the algorithms involved. For new algorithms using parametrized heuristics, similar experiments are invaluable to the usability.

A modular approach for the development of sequential covering algorithms beyond the rule heuristic is surely also recommended (Martens et al, 2011; Montes de Oca et al, 2009). The creation of novel algorithms would benefit from an extensive experimental design which studies the importance and interplay of the metaheuristic variant, objective function to optimize (rule heuristic

[16] The data of these experiments is available online at http://www.antminerplus.com

in this case), parameter settings and data characteristics. Finally, open source implementations would facilitate an efficient and easy incorporation of novel building block instantiations.

We hope to have demonstrated that empirical analysis for novel sequential covering algorithms should go beyond the traditional hyper-parameter tuning. By following the set forth guidelines and experimental setup for tuning the rule heuristic (potentially for each dataset separately) the generalization results can be dramatically improved. To conclude, the accuracy and comprehensibility gains observed in our large scale empirical study provide a strong affirmitive answer to our initial question 'To tune or not to tune'.

Acknowledgment

This work was carried out using the Stevin Supercomputer Infrastructure at Ghent University. We would like to thank the Flemish Research Council for the financial support (FWO Odysseus grant B.0915.09). We are also grateful to the creators of the open source implementations used in this work.

References

Aguilar-Ruiz J, Riquelme JC, Toro M (2003) Evolutionary learning of hierarchical decision rules. IEEE Trans Syst Man Cybern Part B Cybern 33:324–331

Aguilar-Ruiz JS, Giráldez R, Santos JCR (2007) Natural encoding for evolutionary supervised learning. IEEE Trans Evol Comput 11(4):466–479

Alcalá-Fdez J, Sánchez L, García S, del Jesus M, Ventura S, Garrell J, Otero J, Romero C, Bacardit J, Rivas V, Fernández J, Herrera F (2009) Keel: a software tool to assess evolutionary algorithms for data mining problems. Soft Comput 13:307–318

Alcalá-Fdez J, Fernandez A, Luengo J, Derrac J, García S, Sánchez L, Herrera F (2011) Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. J Mult-Valued Logic Soft Comput 17:255–287

An A, Cercone N (2000) Rule quality measures improve the accuracy of rule induction: An experimental approach. In: Proc 12th Int Symp Found Intell Syst, ISMIS'00, pp 119–129

Andrews R, Diederich J, Tickle AB (1995) Survey and critique of techniques for extracting rules from trained artificial neural networks. Knowl-Based Syst 8(6):373–389

Baesens B, Setiono R, Mues C, Vanthienen J (2003a) Using neural network rule extraction and decision tables for credit-risk evaluation. Manag Sci 49(3):312–329

Baesens B, Van Gestel T, Viaene S, Stepanova M, Suykens J, Vanthienen J (2003b) Benchmarking state-of-the-art classification algorithms for credit scoring. J Oper Res Soc 54(6):627–635

Cestnik B (1990) Estimating probabilities: A crucial task in machine learning. In: 9th Eur Conf Artif Intell, ECAI'90, pp 147–149

Clark P, Niblett T (1989) The CN2 induction algorithm. Mach Learn 3:261–283

Cohen W (1995) Fast effective rule induction. In: Proc 12th Int Conf Mach Learn, ICML'95, pp 115–123

Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30

Dorigo M, Maniezzo V, Colorni A (1996) Ant System: Optimization by a colony of cooperating agents. IEEE Trans Syst Man Cybern Part B Cybern 26(1):29–41

Fayyad UM, Irani KB (1992) On the handling of continuous-valued attributes in decision tree generation. Mach Learn 8(1):87–102

Fernández A, García S, Luengo J, Bernadó-Mansilla E, Herrera F (2010) Genetics-based machine learning for rule induction: state of the art, taxonomy, and comparative study. IEEE Trans Evol Comput 14:913–941

Freitas AA (2003) A survey of evolutionary algorithms for data mining and knowledge discovery. In: Ghosh A, Tsutsiu S (eds) Advances in evolutionary computing: theory and applications, Springer-Verlag New York, Inc, New York, NY, USA, pp 819–845

Fürnkranz J (1999) Separate-and-conquer rule learning. Artif Intell Rev 13(1):3–54

Fürnkranz J, Flach P (2005) ROC 'n' rule learning - towards a better understanding of covering algorithms. Mach Learn 58(1):39–77

Fürnkranz J, Flach PA (2003) An analysis of rule evaluation metrics. In: Proc 20th Int Conf Mach Learn, ICML'03, pp 202–209

García S, Herrera F (2008) An extension on statistical comparisons of classifiers over multiple data sets for all pairwise comparisons. J Mach Learn Res 9:2677–2694

Hall M, Holmes G (2003) Benchmarking attribute selection techniques for discrete class data mining. IEEE Trans Knowl Data Eng 15(6):1437–1447

Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH (2009) The weka data mining software: an update. SIGKDD Explor Newsl 11:10–18

Hettich S, Bay SD (1996) The uci kdd archive [http://kdd.ics.uci.edu]

Holden N, Freitas A (2005) A hybrid particle swarm/ant colony algorithm for the classification of hierarchical biological data. In: Proc IEEE Swarm Intell Symp, SIS'05, pp 100–107

Holden N, Freitas AA (2008) A Hybrid PSO/ACO Algorithm for Discovering Classification Rules in Data Mining. Journal of Artificial Evolution and Applications 2008:1–12

Janssen F, Fürnkranz J (2009) A re-evaluation of the over-searching phenomenon in inductive rule learning. In: Proc SIAM Int Conf Data Min, SDM'09, pp 329–340

Janssen F, Fürnkranz J (2010) On the quest for optimal rule learning heuristics. Mach Learn 78(3):343–379

Kira K, Rendell L (1992) The feature selection problem: Traditional methods and a new algorithm. In: Proc 10th Natl Conf Artif Intell, AAAI'92, pp 129–134

Klösgen W (1992) Problems for knowledge discovery in databases and their treatment in the statistics interpreter EXPLORA. Int J Intell Syst 7(7):649–673

Kononenko I (1994) Estimating attributes: Analysis and extensions of RELIEF. In: Eur Conf Mach Learn, ECML'94, pp 171–182

Lessmann S, Baesens B, Mues C, Pietsch S (2008) Benchmarking classification models for software defect prediction: A proposed framework and novel findings. IEEE Trans Software Eng 34(4):485–496

Liu B, Abbass HA, McKay B (2002a) Density-based heuristic for rule discovery with ant-miner. In: Proc 6th Australasia-Japan Joint Workshop Intell Evol Syst, AJWIS'02, pp 180–184

Liu B, Abbass H, McKay B (2003) Classification rule discovery with ant colony optimization. In: Proc IEEE/WIC Int Conf Intell Agent Tech, IAT'03, pp 83–88

Liu H, Hussain F, Tan CL, Dash M (2002b) Discretization: An enabling technique. Data Min Knowl Discov 6:393–423

Lopes HS, Coutinho MS, Lima WC (1997) An evolutionary approach to simulate cognitive feedback learning in medical domain. In: Sanchez E, Shibata T, Zadeh L (eds) Genetic Algorithms and Fuzzy Logic Systems: Soft Computing Perspectives, World Scientific, Singapore, pp 193–207

Martens D, Baesens B, Van Gestel T, Vanthienen J (2007a) Comprehensible credit scoring models using rule extraction from support vector machines. Eur J Oper Res 183(3):1466–1476

Martens D, De Backer M, Haesen R, Vanthienen J, Snoeck M, Baesens B (2007b) Classification with ant colony optimization. IEEE Trans Evol Comput 11(5):651–665

Martens D, Baesens B, Fawcett T (2011) Editorial survey: Swarm intelligence for data mining. Mach Learn 82(1):1–42

Novak PK, Lavrač N, Webb GI (2009) Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup mining. J Mach Learn Res 10:377–403

Montes de Oca M, Stützle T, Birattari M, Dorigo M (2009) Frankenstein's PSO: A composite particle swarm optimization algorithm. IEEE Trans Evol Comput 13(5):1120–1132

Otero FEB, Freitas AA, Johnson CG (2009) Handling continuous attributes in ant colony classification algorithms. In: Proc IEEE Symp Comput Intell Data Min, IEEE, CIDM'09, pp 225–231

Parpinelli R, Lopes H, Freitas A (2002) Data mining with an ant colony optimization algorithm. IEEE Trans Evol Comput 6(4):321–332

Parpinelli RS, Lopes HS, Freitas AA (2001) An ant colony based system for data mining: Applications to medical data. In: Proc Genet Evol Comput Conf, GECCO'01, pp 791–797

Pazzani M, Mani S, Shankle W (2001) Acceptance by medical experts of rules generated by machine learning. Methods of Information in Medicine 40(5):380–385

Quinlan J (1993) C4.5 Programs for Machine Learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA

Quinlan JR (1990) Learning logical definitions from relations. Mach Learn 5:239–266

van Rijsbergen CJ (1979) Information Retrieval. Butterworth-Heinemann, Newton, MA, USA

Salama K, Abdelbar A (2011) Exploring different rule quality evaluation functions in aco-based classification algorithms. In: IEEE Symp Swarm Intell, SIS'11, pp 1–8

Sousa T, Silva A, Neves A (2004) Particle swarm based data mining algorithms for classification tasks. Parallel Comput 30(5-6):767–783

Steuer R (1986) Multiple Criteria Optimization: Theory, Computation and Application. John Wiley, New York

Stützle T, Holger HH (2000) MAX-MIN ant system. Future Generat Comput Syst 16:889–914

Suykens JAK, Van Gestel T, Brabanter JD, De Moor B, Vandewalle J (2002) Least Squares Support Vector Machines. World Scientific, Singapore

Tan PN, Kumar V, Srivastava J (2002) Selecting the right interestingness measure for association patterns. In: Proc 8th ACM SIGKDD Int Conf Knowl Discov Data Min, KDD'02, pp 32–41

Tan PN, Steinbach M, Kumar V (2005) Introduction to Data Mining. Addison Wesley, Boston, MA

Van Gestel T, Suykens J, Baesens B, Viaene S, Vanthienen J, Dedene G, De Moor B, Vandewalle J (2004) Benchmarking least squares support vector machine classifiers. Mach Learn 54:5–32

Venturini G (1993) SIA: a supervised inductive algorithm with genetic search for learning attributes based concepts. In: Proc Eur Conf Mach Learn, ECML'93, pp 280–296

Verbeke W, Martens D, Mues C, Baesens B (2011) Building comprehensible customer churn prediction models with advanced rule induction techniques. Expert Systems with Applications 38(3):2354–2364

Witten IH, Frank E (2005) Data Mining: Practical Machine Learning Tools and Techniques, 2nd edn. Morgan Kaufmann, San Francisco, CA, USA

Wrobel S (1997) An algorithm for multi-relational discovery of subgroups. In: Proc 1th Eur Symp Princ Data Min Knowl Discov, PKDD'97, pp 78–87