Combinatorial optimization in inductive concept learning

Dunja Mladenić
Artificial Intelligence Laboratory
Computer Science Department
J. Stefan Institute, Jamova 39
61000 Ljubljana, Slovenia
dunja.mladenic@ijs.si

Abstract

Local optimization algorithms, standardly used in combinatorial optimization, can be applied in inductive concept learning. Learning can be defined as search of the space of concept descriptions, guided by some heuristic function. The paper presents learning with stochastic local optimization algorithms (based on *Simulated annealing*) and deterministic local optimization algorithms (*k-opt* known from solving the 'travelling salesman' problem). These algorithms and some of their combinations have been tested within the ATRIS shell, and their performance compared on different real-world machine learning problems. The rule induction shell ATRIS was developed to enable simple use of existing and adding of new optimization algorithms and noise-handling mechanisms, to enable simple 'cross validation' experiments, experiments with different attributes considered as class and different attribute subsets used for learning. The main objective of this paper is an empirical analysis of different optimization algorithms and some of their combinations in comparison with a decision tree learning algorithm.

1 INTRODUCTION

Inductive concept learning can be viewed as search of the space of concept descriptions. The choice of the description language of concepts L_C determines the search space, called the *hypothesis space.* Depending on the choice of the hypothesis language, *inductive learning systems* can learn either attribute descriptions or first-order relational descriptions. We will only relate to *attribute-value learning*, in which training examples and induced hypotheses are described in terms of a fixed collection of attribute values.

In inductive learning systems, the structure of the search space is determined by generalisation/specialisation operators. Usually locally optimal search is performed by using a hill-climbing and/or a beam-search strategy. For example, the decision tree induction system ASSISTANT (Cestnik, Kononenko and Bratko 1987) performs hill-climbing, and the rule induction systems AQ15 (Michalski et al. 1986) and CN2 (Clark and Niblett 1989, Clark and Boswell 1991) perform beam-search. Their success in dealing with real-world problems lies also in the choice of *heuristics* used to guide the search and to stop the construction of trees/rules.

Local optimization algorithms, standardly used in combinatorial optimization, can be applied in inductive concept learning. In general, local combinatorial optimization (Papadimitriou and Steiglitz 1982) is based on: (1) a *set of candidate solutions* of a given problem Φ, (2) a *cost function* $P : \Phi \rightarrow \Re$ used to guide the search for an optimal solution and (3) the *transformations* used for 'moving' from one solution to another. To apply known local optimization algorithms in inductive concept learning, three above mentioned categories must be defined: (1) is the hypothesis space; (2) is some heuristic function; (3) are the transformations that depend on the internal concept description representation .

The paper presents the use of stochastic local optimization algorithms (based on *Simulated annealing*) (Press et al. 1987) and deterministic local optimization algorithms (*k-opt*, known from solving the 'travelling salesman' problem) (Syslo, Deo and Kowalik 1983). Algorithms and some of their combinations have been tested and compared on different real-world machine learning problems. Section 2 introduces the concept learning problem and the concept description language. Section 3 describes each of the combinatorial optimization algorithms used in our experiments. Section 4 presents the rule induction shell ATRIS inside which the combinatorial optimization algorithms were used. The results of the experiments are given in Section 5. Section 6 concludes with some ideas for the further work.

2 PROBLEM DEFINITION

In learning from examples, the *target concept* C is a subset of the given *example space* of objects, labelled $\oplus$ if they are elements of the target concept (*positive examples*), and $\ominus$ (*negative examples*) otherwise. Instead of being classified into one of the two *classes* ($\oplus$ or $\ominus$), an object can be classified into one of the mutually exclusive classes C_k, $k = 1, \ldots, n$. The learning task is defined as follows: Given is the description language of examples L_E, the hypothesis language L_C, the coverage relation between L_C and L_E and the quality criterion defined on elements of L_C. The task is to find a *concept description*, expressible in L_C, satisfying the quality criterion. Usually, the quality criterion requires the induced hypothesis to be complete and consistent with respect to the examples, i.e., that it covers all the positive examples and no negative example. When learning from imperfect data, the consistency and completeness criteria need to be relaxed in order to avoid overly specific hypotheses (see stopping criteria in the Section 4.2.1).

Let L_E be the language of attribute value tuples, and L_C a language of *if-then* rules. In an attribute-value hypothesis language, a *selector* relates an attribute to a value or a disjunction (so-called *internal disjunction*) of values. As an example from a simple 'Saturday morning' domain (Quinlan 1986), $[Outlook = sunny \vee rainy]$ is a selector. A conjunction of selectors forms a *complex*:

$$R_i = \bigwedge_j \left[A_j = \bigvee_k V_{jk} \right]$$

R_i stands for the i-th *complex*, A_j for the j-th attribute and V_{jk} for the k-th value of the j-th attribute. A hypothesis is a set of *if-then* rules of the form **if** *Complex* **then** *Class*. An example rule is the following:
if $[Outlook = sunny] \bigwedge [Humidity = normal]$ **then** $Suitable = yes$.

In the ATRIS rule induction shell, briefly described in Section 4, an internal bit representation is used, similar to the one used by De Jong and Spears (De Jong and Spears 1991). For example, the previously mentioned Quinlan's 'Saturday morning' domain has four attributes:
A1 - Outlook $\in$ {*sunny, overcast, rainy*}, *A2 - Temperature* $\in$ {*hot, mild, cool*}, *A3 - Humidity* $\in$ {*high, normal*}, *A4 - Windy* $\in$ {*no, yes*} . An example complex

$$[Outlook = sunny] \bigwedge [Humidity = normal]$$

is represented with the following binary vector:

A_1			A_2			A_3		A_4	
1	0	0	0	0	0	0	1	0	0

3 COMBINATORIAL OPTIMIZATION ALGORITHMS USED FOR RULE INDUCTION

In ATRIS, inside the covering algorithm (described in Section 4.2.1) for a single rule induction, some local combinatorial optimization algorithm is used. The two stochastic search algorithms, *Simulated annealing* (Press et al. 1987) and *'Markovian neural networks'* (Kovačič 1991), are described in Section 3.1. Deterministic search algorithms *k-opt* are described in Section 3.2. Notice that for each algorithm at least one *transformation* used for 'moving' from one candidate solution to another should be defined. With the transformations, the neighbourhood for each candidate solution (concept description) is defined. In all used local combinatorial optimization algorithms the start position has been randomly chosen.

3.1 STOCHASTIC LOCAL SEARCH ALGORITHMS

The method of *Simulated annealing* is based on the analogy with thermodynamics, (for slowly cooled systems, nature is able to find the minimum energy state). For a selected evaluation function, which is to be minimized, the method performs stochastic search in a discrete state space. To apply this algorithm to the optimization of a rule represented as a binary vector, the modification (negation) of a randomly chosen bit in the rule binary vector, is taken as the *transformation*. For example, neighbour of the rule (1 0 0 0 0 0 0 1 0 0) from Section 2 is the rule (0 0 0 0 0 0 0 1 0 0) if the first value was selected for modification, or the rule (1 1 0 0 0 0 0 1 0 0) if the second value was chosen, etc.

Simulated annealing algorithm applied to rule optimization is as follows:

$T :=$ StartTemperature;

```
current_rule := random rule; {random binary
    vector}
repeat
    Randomly select value in the current_rule bi-
    nary vector;
    if modification of the selected value improves
    the current_rule then
        current_rule := modified_rule;
    else {modified rule is of lower quality}
        With probability exp((qmr − qr)/T) exe-
        cute:
            current_rule := modified_rule;
    end if
    T := T * factor; {factor < 1}
until T < EndTemperature;
output current_rule;
```

Parameter T (temperature) is used in the probability function[1] and to control the **repeat-until** loop. Probability function, used in the above algorithm, is calculated from the quality of a modified rule (qmr), the quality of the current rule (qr) and the parameter temperature (T).

The other stochastic search method is an extension of simulated annealing based on 'Markovian neural networks' (Kovačič 1991) which perform 'Markovian' simulated annealing. The difference is that we randomly select more than one value in the binary vector and probabilisticaly choose one to modify the rule; better modifications (in terms of rule quality) are more likely to be chosen.

3.2 DETERMINISTIC LOCAL SEARCH ALGORITHMS

Deterministic search algorithms named *k-Opt*, can be modified for the purpose of a single rule optimization as listed below. The *transformation* used in these algorithms for 'moving' in the candidate solution space is the modification (negation) of *1, 2, ..., k*, randomly chosen bits in a binary vector representing a rule.

```
current_rule := random rule; {random binary
    vector}
repeat
    Find one (two, or three) values in the rule bi-
    nary vector, modification of which maximizes
    the quality of the rule (i.e., no other modifi-
    cation of one (two, or three) values provides
    a rule with better quality);
    if modified_rule is of better quality than
    current_rule then
        current_rule := modified_rule;
    end if
until no modification to better rule quality is
    possible;
output current_rule;
```

[1] The lower the temperature, the more unlikely the significant change to lower quality rule.

The above algorithm is called *1-Opt* if (in the first step of **repeat-until** loop) it checks only rules with one modified value, *2-Opt* if it checks rules with one or two modified values, *3-Opt* if it checks rules with one, two or three modified values. In general *k-Opt* can be preformed by checking rules with up to k values modified. The 'faster' variant of algorithm (*2-OptF*, *3-OptF*) searches for the first modification that provides rule with better quality and not the modification which maximizes quality of the rule.

In the ATRIS shell, currently *2-Opt*, *3-Opt*, *2-OptF* or *3-OptF* can be used for learning and *1-Opt*, *2-Opt*, *3-Opt*, *2-OptF* or *3-OptF* for post-processing.

4 ATRIS: A SHELL FOR RULE INDUCTION

The idea of the ATRIS shell is to enable simple design of experiments, usage of existing (and simple adding of new) optimization algorithms and heuristic functions. When different algorithms are integrated in a single system (shell), the idea of combining them comes naturally. The ATRIS rule induction shell supports three main parts: pre-processing of data, construction of a hypothesis and its post-processing.

4.1 PRE-PROCESSING

The input to ATRIS is designed to enable a simple design of experiment with different parameters (selected heuristic, learning algorithm, post-processing algorithm, example validation). Additionally, different experiments can be performed on the same set of examples by selecting an attribute to be treated as the class and the attributes to be used in the induced hypothesis. The pre-processor of ATRIS allows for undefined attribute values in examples (in the current implementation these are simply replaced by the majority attribute value) and the use of built-in background knowledge. For example, the pre-processor can check for the equality of attribute values on pairs of attributes of the same type (having the same set of possible values). Using this idea, originating from the LINUS relation learning system (Lavrač, Dzeroski and Grobelnik 1991), an extended set of tuples is generated and used in learning.

Pre-processed data are converted into an internal representation described in Section 2.

4.2 THE HYPOTHESIS CONSTRUCTION

4.2.1 The basic covering algorithm

The construction of a hypothesis is performed by combinatorial optimization algorithms used within the classical covering algorithm as follows:

```
Hypothesis := ∅;
for each class do
   PExs := positive training examples;
   NExs := negative training examples;
   repeat
      Rule := new rule induced from PExs and
     NExs; {where induction is perfomed by a
     combinatorial optimization algorithm}
      Hypothesis := Hypothesis ⋃ Rule;
      From PExs delete examples covered by
     Rule;
   until PExs = ∅ (or some other stopping cri-
     terion is satisfied);
end for
output Hypothesis;
```

The covering algorithm loop is controlled by a *stopping criterion*, used to decide when to stop the induction of rules for the current class. This criterion can be referred to as the *sufficiency criterion*. Combinatorial optimization algorithms contain a *stopping criterion*, usually referred to as the *necessity criterion*, if it is part of the heuristic function that is used to guide the search. In domains with exact and non-noisy data, the sufficiency criterion requires completeness (coverage of all positive examples) and the necessity criterion requires consistency (no covered negative example). In domains with imperfect data, the two criteria are implemented as heuristics for evaluating the expected classification accuracy, aimed at avoiding overly specific hypotheses.

4.2.2 Heuristic function

Several probability estimates can be applied to evaluate the quality (the classification accuracy) of a rule. In the ATRIS shell, four heuristic functions are implemented, the *relative frequency* being the simplest, but also unreliable when estimating probabilities from a small training set. To alleviate this problem, the *Laplace estimate* can be used. This estimate, however, relies on the assumption that the prior probability of the classes is uniformly distributed (Mingers 1989), which is rarely true in practice. To avoid this assumption, Cestnik (Cestnik 1990) proposed the *m-estimate* which was successfully used in the 'naive' Bayesian classifier (Cestnik 1990) and in tree pruning (Cestnik and Bratko 1991). The parameter m used in the m-estimate is set subjectively and expresses our confidence in the experimental evidence (training examples). In practical use, it was found that setting the value for parameter m is not a trivial problem. Quinlan (Quinlan 1991) uses an *adapted form of the Bayes-Laplace* formula, with the aim to improve the estimate for the accuracy of small disjuncts. Namely, Quinlan offers a formula that uses the prior probability of the target class in the 'context' of the rule (and no other parameter like m); "although no theoretical justification for this choice can be given" (Quinlan 1991).

Since the aim of this paper is to analyse different local optimization algorithms and their possible combinations, the simplest heuristic function (*relative frequency*) was used in the experiments.

4.3 POST-PROCESSING

The main post-processing mechanism in ATRIS uses deterministic local optimization algorithms (described in Section 3.2) to refine each induced rule. This mechanism is briefly described below. Another mechanism is a variant of *reduced error pruning* (Quinlan 1987) which enables deletion of redundant rules.

The deterministic local optimization post-processing mechanism can be applied to improve the results of both stochastic and deterministic optimization algorithms as follows.

- The stochastic search algorithm typically finds a solution which can still be locally optimized, since search is stopped by the value of the parameter called 'temperature' (see Section 3.1). For example, the results of learning with Simulated annealing can be optimized by post-processing with *3-Opt* (see Section 5).

- The deterministic search algorithm finds locally optimal solution (see Section 3.2) and post-processing has to be performed by some 'wider' (lager neighbourhood) deterministic optimization algorithm. For example, learning with *2-Opt* algorithm and post-processing with *3-Opt* algorithm give typically better results than just learning with *2-Opt*, and uses less time than just learning with *3-Opt* (see Section 5).

5 EXPERIMENTS AND RESULTS

To investigate the combinations of local optimization algorithms implemented in ATRIS, an empirical analysis of implemented algorithms was performed. All experiments were performed on a SUN-4. Experiments were carried out on three machine learning domains. For each domain, 10 validations on the 70% training—30% testing split were used; for each validation, four experiments were performed with a different initial value for the random number generator (Park and Miller 1988).

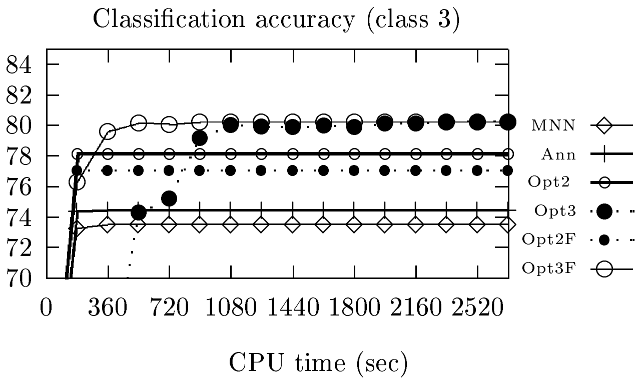

Figure 1: Classification accuracy on testing examples when focusing rules for class 3, domain 'lymphography'.

Domain used for the first experiments is the 'lymphography' medical domain, consisting of 148 examples describing lymphographic symptoms with 18 attributes and classified into one of 4 diagnostic classes (Cestnik, Kononenko and Bratko 1987, Clark and Boswell 1991). Figure 1 presents classification accuracy for the third class (on testing examples) of six local optimization algorithms: *Simulated annealing*, *Markovian neural networks*, *2-Opt*, *3-Opt*, *2-OptF*, *3-OptF*.

When calculating the classification accuracy, some examples can be covered by different rules describing different classes. In such case, Atris uses the voting method (Kononenko 1991) to predict the class of the example. On the other hand, some of the examples may not be covered by the induced rules. Atris uses a very simple method to overcome this problem; to each uncovered example the majority class is assigned.

For the purpose of the algorithm analysis, classification accuracy was calculated at different *CPU time limits* (e.g. 180 sec, 360 sec, ...).[2] When classifying testing examples for a specific *time limit* a *focused* class (e.g. class 3 in the Figure 1) is selected. For the focused class, only the rules induced in the time up to *time limit* (e.g. in the first 360 sec) were used in the classification. For each other class, all the induced rules were used (induced in any time limit). For example, in Figure 1 for the *time limit* 360 sec *Markovian neural networks* achieved 73.5 % accuracy and *3-OptF* achieved 80.2 % accuracy.

From Figure 1 one is inclined to divide the used algorithms, according to the achieved accuracy, into three groups: (1) stochastic algorithms (the lowest accuracy), (2) *2-Opt* and *2-OptF*, and *3-Opt* and *3-OptF* (the highest accuracy). Furthermore, an analysis of the (increasing) number of induced rules and the (decreasing) number of uncovered positive training examples in CPU time was performed. The result was the idea to combine stochastic algorithms (for searching rules) and deterministic algorithms (to perform locally optimal search in post-processing of rules) and to combine two deterministic algorithms (the faster for searching rules and the slower but wider for post-processing)[3]. Stochastic algorithms, *Simulated annealing* (Ann) and *Markovian neural networks* (MNN), were combined with deterministic algorithms, *1-Opt*(Opt1), *3-Opt*(Opt3), *3-OptF*(Opt3F). *Opt-1* was used for post-processing as the simplest and *3-Opt* and *3-OptF* as the most accurate ones. Deterministic algorithms were tested with and without post-processing.

For the analysis of 12 algorithm combinations (see Figure 2), three machine learning domains were used: 'lymphography' already described above; 'primary tumor', also a medical domain consisting of 339 examples and describing location of primary tumor with 17 attributes and 22 diagnostic classes; 'KRK' (chess endgame positions) domain consisting of 1000 examples describing engame positions with 6 attributes and 2 classes (whether win for black). The number of induced rules grows linearly with the CPU time and is similar for different algorithms except in domain 'lymphography', where *Simulated annealing* in combination with *1-Opt* (AnnOpt1) induced twice as many rules as the most of the other methods, but also achieved a very low accuracy (see Figure 2). Classification accuracy is presented at Figure 2. For all domains, algorithms 'group' with respect to CPU time used. The fastest were *Simulated annealing* (AnnOpt1), *2-Opt* (Opt2), *2-OptF* (Opt2F) and the slowest *3-Opt* (OPt3). The highest classification accuracy was achieved by combinations of algorithms using *3-Opt* or *3-OptF* in post-processing, e.g., by MNNOpt3 and Opt2FOpt3F, MNNOpt3F, MNNOpt3F and AnnOpt3F in the three domains respectively.

To get an idea about accuracy of the analysed algorithms, experiments with theAssistant algorithm were performed on the same training/testing data. Assistant, using no pruning mechanisms, achieved 77.24% accuracy on 'lymphography', 39.95% on 'pri-

[2]For each class, time was measured separately. Each rule is then labelled with the time used for its induction.

[3]Notice that *2-Opt* is in some way a subset of *3-Opt*.

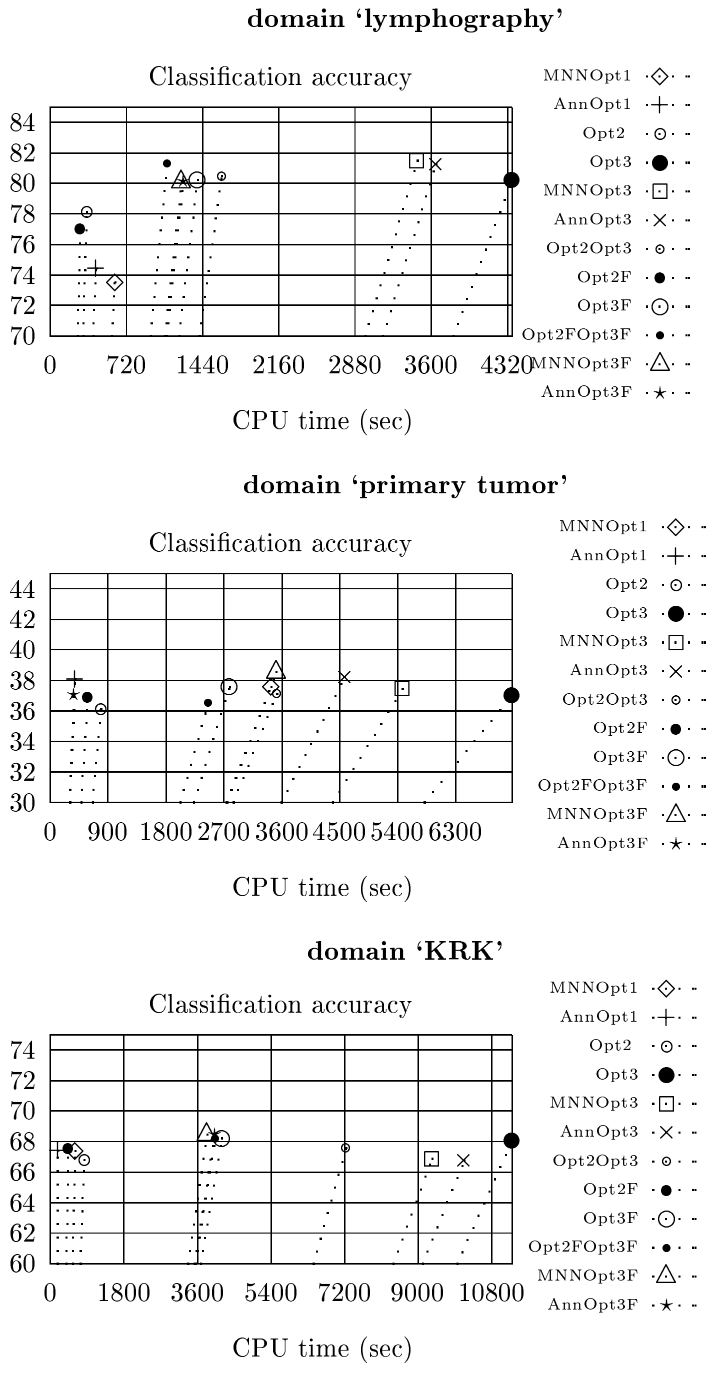

Figure 2: The analysis of optimization algorithms for three domains (classification of test examples).

mary tumor' and 64.35% on 'KRK'. Most of the analysed optimization algorithms achieved as good or even better accuracy than ASSISTANT[4].

6 CONCLUSION AND FURTHER WORK

Stochastic and deterministic local optimization algorithms, standardly used in combinatorial optimization, have been applied in inductive concept learning. Analysis of these optimization algorithms was performed on three well known machine learning domains. To enable simple testing of different optimization algorithms and simple design of experiments, the rule induction shell ATRIS was developed. At the moment it has six optimization algorithms and four heuristic estimates. Results of the experiments inspired the idea of combining the optimization algorithms to achieve an efficiency/accuracy tradeoff. Some algorithm combinations were tested by means of learning with one and post-processing with the other, while rule induction with two (or more) different optimization algorithms is left for further work.

In further work we plan to improve the noise-handling and classification methods, such as dealing with undefined attribute values, prediction of the class for an uncovered example, combining of rules covering the same classification example. It shoud be noted that in the experiments the simplest heuristic for estimating the expected classification accuracy was used. Certainly, better accuracy could have been achieved by using the other three heuristics already implemented. Some further improvement of results could be achieved by adding new optimization algorithms in the ATRIS shell (as *Tabu search* (Glover 1990)), as well as by using existing and adding new built-in background knowledge. The main idea which was provided by the experiments is that learning should be performed with some accurate and quite fast algorithm (e.g. *Markovian neural networks* post-processed with *3-OptF*) using an appropriate heuristic estimate considering noise in the specific domain. At the moment, using the ATRIS capability of choosing the attribute to be treated as the class and the attributes to be used in the hypothesis, learning of rules describing functional dependencies among attributes is being investigated (Savnik 1993).

[4]Notice that slightly lower accuracy achieved on 'primary tumor' could be a consequence of the fact that ATRIS assigns the majority attribute value to examples with unknown values and does not use the more sophisticate 'naive' Bayesian formula as ASSISTANT does.

Acknowledgements

This work was supported by the Slovenian Ministry of research and Technology. Medical data sets were obtained from the University Medical Center in Ljubljana, provided by Milan Sokolič ('lymphography') and Matjaž Zwitter ('primary tumor'). Chess endgame ('KRK') data set was originaly provided by Michael Bain. I wish to thank Marko Grobelnik, Nada Lavrač and Darko Zupanič for their contribution in the first phase of development of the ATRIS shell (Mladenić et al. 1992) and Ivan Bratko, Nada Lavrač, Ross Quinlan and Irma Sutlič for their comments to the paper.

References

Cestnik, B. (1990) Estimating probabilities: A crucial task in machine learning. *Proc. of ECAI'90*, pp. 147-149, Stockholm, Sweden.

Cestnik, B., Kononenko, I. and Bratko, I. (1987) ASSISTANT 86: A knowledge elicitation tool for sophisticated users. In Bratko, I. and Lavrač, N. (eds.) *Progress in machine learning*, pp. 31-45. Wilmslow, Sigma Press.

Cestnik, B. and Bratko, I. (1991) On estimating probabilities in tree pruning. *Fifth European Working Session on Learning, EWSL 91*, pp. 138-150. Porto, Portugal, Springer-Verlag.

Clark, P. and Niblett, T. (1989) The CN2 induction algorithm. *Machine Learning 3 (4)*, pp. 261-283. Kluwer Academic Publishers.

Clark, P. and Boswell, R. (1991) Rule induction with CN2: Some recent improvements. *Proc. of EWSL 91*, pp. 151-163. Porto, Portugal, Springer-Verlag.

De Jong, K.A. and Spears, W.M. (1991) Learning concept classification rules using genetic algorithms. *Proc. IJCAI-91*, pp. 651-656. Sydney, Australia. Morgan Kaufmann.

Glover, F. (1990) Tabu search – part I/II. *ORSA Journal on Computing 2, 3 (1)*, pp. (I) 190-206, (II) 4-32.

Kononenko, I. (1991) An experiment in machine learning of redundant knowledge. *Proc. of Intern. Conf. MELCON 1991*, pp. 1146-1149. Ljubljana, Slovenia.

Kovačič, M. (1991) Markovian neural networks. *Biological Cybernetics 64*, pp. 337-342. Springer-Verlag.

Lavrač, N., Džeroski, S. and Grobelnik, M. (1991) Learning nonrecursive definitions of relations with LINUS. *Proc. of EWSL 91*, pp. 265-281. Porto, Portugal, Springer-Verlag.

Michalski, R.S., Mozetič, I., Hong, J. and Lavrač. N. (1986) The multipurpose incremental learning system AQ15 and its testing application to three medical domains. *Proc. of AAAI-86*, pp. 1041-1045. PA: Philadelphia. Morgan Kaufmann.

Mingers, J. (1989) An empirical comparison of pruning methods for decision tree induction. *Machine Learning 4 (2)*, pp. 319-342. Kluwer Academic Publishers.

Grobelnik, M. and Lavrač, N. (1992) Stochastic search in inductive concept learning. *Proc. of ISSEK Workshop'92.* Bled, Slovenia.

Papadimitriou, C.H. and Steiglitz, K. (1982) Optimization problems. In *Combinatorial optimization: Algorithms and complexity*, pp. 1-24. Prentice-Hall, Inc. Englewood Cliffs

Park, S.K. and Miller, K.W. (1988) Random number generators: good ones are hard to find. *Comm. ACM 31, 10*, pp. 1192-1201.

Press, W.H., Flannery, B.P., Teukolsky, S.A. and Vetterling, W.T. (1987) Combinatorial minimization: Method of simulated annealing. In *Numerical Recipes: The art of scientific computing*, pp. 326-334. Cambridge University Press.

Quinlan, J.R. (1986) Induction of decision trees. *Machine Learning 1*, pp. 81-106. Kluwer Academic Publishers.

Quinlan, J.R. (1987) Simplifying decision trees. *International Journal of Man-Machine Studies 27*, pp. 221-234. Academic Press.

Quinlan, J.R. (1991) Improved estimate for the accuracy of small disjuncts, Technical note.*Machine Learning 6*, pp. 93-98. Kluwer Academic Publishers.

Savnik, I. (1993) Induction of functional dependencies from relations.*Technical report*, J.Stefan Institute, Ljubljana, Slovenia.

Syslo, M.M., Deo, N. and Kowalik, J.S. (1983) Local search heuristics. In *Discrete optimization algorithms*, pp. 368-380. Prentice-Hall. Inc. Englewood Cliffs