Advances in Applying Genetic Programming to Machine Learning, Focussing on Classification Problems

Stephan M. Winkler[1,2], Michael Affenzeller[1], and Stefan Wagner[1]

[1] Upper Austrian University of Applied Sciences
College of Information Technology at Hagenberg
Department of Software Engineering
Hauptstraße 117, 4232 Hagenberg, Austria
{stephan, michael, stefan}@heuristiclab.com

[2]Johannes Kepler University Linz
Institute for Design and Control
of Mechatronical Systems
Altenbergerstraße 69, 4040 Linz, Austria
stephan.winkler@jku.at

Abstract

A Genetic Programming based approach for solving classification problems is presented in this paper. Classification is understood as the act of placing an object into a set of categories, based on the object's properties; classification algorithms are designed to learn a function which maps a vector of object features into one of several classes. This is done by analyzing a set of input-output examples ("training samples") of the function. Here we present a method based on the theory of Genetic Algorithms and Genetic Programming that interprets classification problems as optimization problems: Each presented instance of the classification problem is interpreted as an instance of an optimization problem, and a solution is found by a heuristic optimization algorithm. The major new aspects presented in this paper are suitable genetic operators for this problem class (mainly the creation of new hypotheses by merging already existing ones and their detailed evaluation) we have designed and implemented. The experimental part of the paper documents the results produced using new hybrid variants of genetic algorithms as well as investigated parameter settings.

1 Introduction

In general, data mining is understood as the practice of automatically searching large stores of data for patterns. Nowadays, incredibly large (and quickly growing) amounts of data are collected in commercial, administrative, and scientific databases. Several sciences (e.g., molecular biology, genetics, astrophysics, and many others) produce extreme amounts of information which are often collected automatically. This is why it is impossible to analyze and exploit all these data manually; what is needed are intelligent computer systems that can extract useful information (such as general rules or interesting patterns) from large amounts of observations. In short, "data mining is the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data" [4].

Classification is understood as the act of placing an object into a set of categories, based on the object's properties. Objects are classified according to an (in most cases hierarchical) classification scheme also called taxonomy. Amongst many other possible applications, examples of taxonomic classification are biological classification (the act of categorizing and grouping living species of organisms), medical classification and security classification (where it is often necessary to classify objects or persons for deciding whether a problem might arise from the present situation or not). A statistical classification algorithm is supposed to take feature representations of objects and map them to a special, predefined classification label. Such classification algorithms are designed to learn (i.e. to approximate the behavior of) a function which maps a vector of object features into one of several classes; this is done by analyzing a set of input-output examples ("training samples") of the function. Since statistical classification algorithms are supposed to "learn" such functions, we are dealing with a specific subarea of *Machine Learning* and, more generally, *Artificial Intelligence.*

The work described in this paper was done within the Translational Research Project L282 "GP-Based Techniques for the Design of Virtual Sensors" sponsored by the Austrian Science Fund (FWF).

There are several approaches which are nowadays used for solving data mining and, more specifically, classification problems. The most common ones are (as for example described in [12]) decision tree learning, instance-based learning, inductive logic programming (such as Prolog, e.g.) and reinforcement learning.

Unlike these methods, the approach we have designed is a genetic programming (GP) model including appropriate crossover and mutation operators for this problem. This GP approach, described in Section 3, has also been implemented as a part of the already existing “HeuristicLab”, a framework for prototyping and analyzing optimization techniques (developed by Wagner and Affenzeller, described in [13]) for which both generic concepts of evolutionary algorithms and many functions to evaluate and analyze them are available. The programming language chosen for this project (and the HeuristicLab) is C# using the Microsoft .NET Framework 1.1. We have also tested our approach intensively. Moreover, in addition to standard GP implementations, new generic concepts [1], based on evolutionary algorithms and developed to increase the quality of the produced solutions, were used and compared to the classical GP approach. Examples of the results of our test series and an overview of the operators and parameters used are given in Section 4.

2 Evolutionary Computation: Genetic Algorithms and Genetic Programming

Evolutionary Computing is the collective name for heuristic problem-solving techniques based on the principles of biological evolution, which are natural selection and genetic inheritance. One of the greatest advantages of these techniques is that they can be applied to a variety of problems, ranging from leading-edge scientific research to practical applications in industry and commerce; by now, evolutionary algorithms are in use in various disciplines like optimization, artificial intelligence, machine learning, simulation of economic processes, computer games or even sociology. The forms of evolutionary computation relevant for the work described in this paper are genetic algorithms (GA) and genetic programming. The fundamental principles of GAs were first presented by Holland [8]. A GA works with a set of candidate solutions (also known as individuals) called population. During the execution of the algorithm each individual has to be evaluated, which means that a value indicating the “fitness” or “goodness” is returned by a fitness function. New individuals are created on the one hand by combining the genetic make-up of two solution candidates (this procedure is called “crossover” or “recombination”), producing a new ”child” out of two “parents”, and on the other hand by mutating some individuals, which means that randomly chosen parts of genetic information are changed (normally a minor ratio of the algorithm’s population is mutated in each generation).
Beside crossover and mutation, the third decisive aspect of genetic algorithms is selection. In analogy to biology this is a mechanism also called “survival of the fittest”. As already mentioned, each individual is associated with a fitness value. The individual’s probability to propagate its genetic information to the next generation is proportional to its fitness; the better a solution candidate’s fitness value, the higher the probability that its genetic information will be included in the next generation’s population. This procedure of crossover, mutation and selection is repeated many times (over many generations) until some termination criterion is fulfilled.

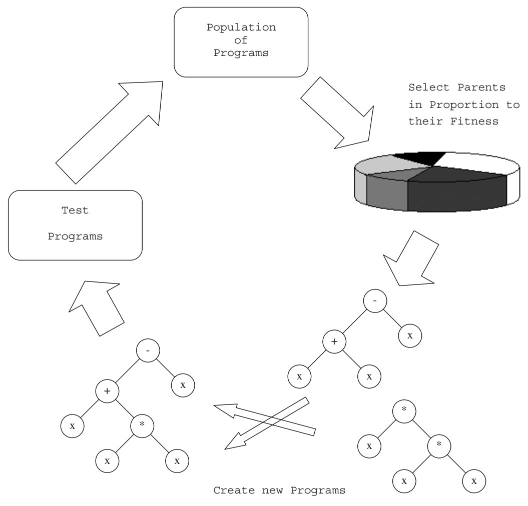

Figure 1. The GP cycle [11].

Genetic Programming was first explored in depth in 1992 by John R. Koza, a computer scientist at Stanford University, CA, USA. In his famous book “Genetic Programming: On the Programming of Computers by Means of Natural Selection” [10] he pointed out that virtually all problems in artificial intelligence, machine learning, adaptive systems, and automated learning can be recast as a search for a computer program, and that genetic programming provides a way to successfully conduct the search for a computer program in the space of computer programs. Similar to GAs, GP works by imitating aspects of natural evolution to generate a solution that maximizes (or minimizes) some fitness function [10]. A population of solution candidates evolves through many generations towards a solution using certain evolutionary operators and a "survival-of-the-fittest" selection scheme. The main difference is that, whereas GAs are intended to find an array of characters or integers representing the solution of a given problem, the goal of a GP process is to produce a computer program (or, as in our case, a formula) solving the optimization problem at hand. Typically the population of a GP algorithm contains a few hundred individuals and evolves through the action of operators known as crossover, mutation and selection. Fig. 1 visualizes how the GP cycle works: As in every evolutionary process, new individuals (in GP's case, new programs) are created. They are tested, and the fitter ones in the population succeed in creating children of their own. Unfit ones die and are removed from the population [11].

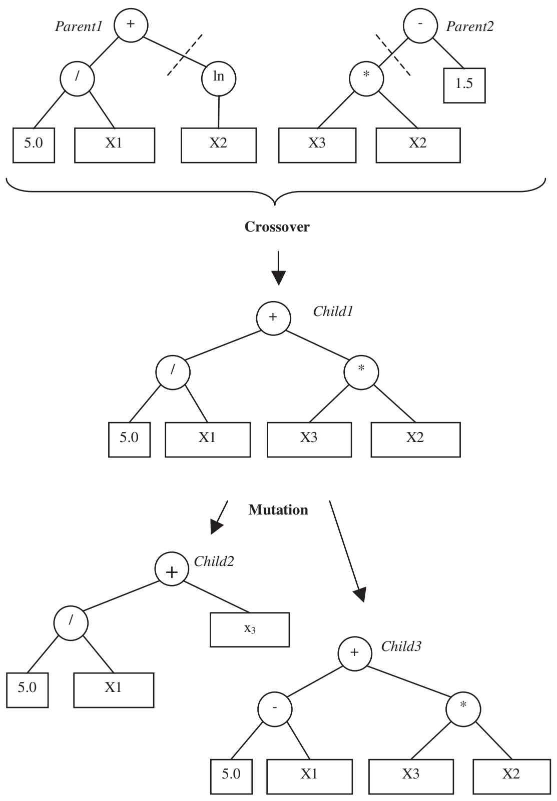

Figure 2. Examples of genetic operations on tree structures: The crossover of parent1 and parent2 yields child1, child2 and child3 are possible mutants of child1.

3 A GP-based classification method

3.1 Representation and Manipulation of Solution Candidates

Preliminary work for the approach presented in this paper was done for the project "Specification, Design, and Implementation of a Genetic Programming Approach for Identifying Nonlinear Models of Mechatronic Systems" which is a part of a bigger strategical project at the Johannes Kepler University Linz, Austria. The goal of this project was to find models for mechatronic systems. It was successfully shown (for instance in [16], and in further detail in [15]) that methods of GP are suitable for determining an appropriate mathematical representation of a physical system. We have used the methods implemented for this project for developing a GP-based approach for a statistical classification algorithm. This algorithm works on a set of training examples with known properties $[X_1 \ldots X_n]$. One of these properties (X_t) has to represent the membership information with respect to the underlying taxonomy. On the basis of the training examples, the algorithm tries to evolve (or, as one could also say, to "learn") a solution, i.e. a formula, that represents the function which maps a vector of object features into one of the given classes. In other words, each presented instance of the classification problem is interpreted as an instance of an optimization problem; a solution is found by a heuristic optimization algorithm.

The goal of the implemented GP classification process is to produce an algebraic expression from a database containing the measured results of the experiments to be analyzed. Thus, in the GP approach we have designed and implemented for this project, the GP algorithm works with solution candidates that are tree structure representations of symbolic expressions. These tree representations consist of nodes and are of variable length. The nodes can either be nonterminal or terminal ones. A nonterminal node signifies a function performing some action on one or more property values within the structure to produce the values of the target property (which of course should be the property which indicates which class the objects belong to). A terminal node represents an input variable (i.e., a pointer to one of the objects' properties) or a constant. The nonterminal nodes have to be selected from a library of possible functions, a pool of potential nonlinear model structural components; the selection of the library functions is an important part of any GP modeling process because this library should be able to represent a wide range of systems. The trees are built by combining nodes according to grammar rules defining the number of inputs for each node type. When the evolutionary algorithm is executed, each individual of the population represents one tree structure; usually the structures are limited by a predefined maximum tree size. Since the tree structures have to be usable by the evolutionary algorithm, mutation and crossover operators for the tree structures have to be designed. Both crossover and mutation processes are applied to randomly chosen branches (in this context a branch is the part of a structure lying below a given point in the tree). Crossing two trees means randomly choosing a branch in each parent tree and replacing the branch of the tree, that will serve as the root of the new child (randomly chosen, too), by the branch of the other tree. Mutation in the context of genetic algorithms means modifying a solution candidate randomly and so creating a new individual. In the case of identifying structures, mutation works by choosing a node and changing it: a function symbol could become another function symbol or be deleted, the value of a constant node or the index of a variable could be modified. This procedure is less likely to improve a specific structure (as extensive test series have shown [15]), but it can help the optimization algorithm to reintroduce genetic diversity in order to restimulate genetic search. An overview of the operators and parameter settings that were used most successfully in the context of structure identification and classification is given in Section 4.

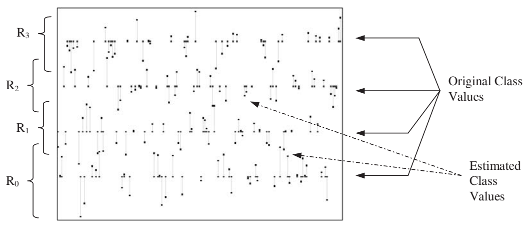

Figure 3. Evaluation of an exemplary classification model including the resulting class value ranges of the given classes.

3.2 Classification Hypotheses Evaluation

Within the approach we present here one also has to deal with the problem that a mathematical formula, which is the output of the classification algorithm, cannot assign a certain class to a given test object. What is done instead is that a value is calculated and, on the basis of this number, one can determine how an object is classified. Furthermore, one then has to determine optimal thresholds between the original class values so that as many objects as possible are classified into the correct class (cf. statistical approaches such as logistic regression). Since the GP algorithm maximizes or minimizes some objective fitness function (better model structures evolve as the GP algorithm minimizes the fitness function), every solution candidate has to be evaluated. In the context of classification this function should be an appropriate measure of the level of agreement between the calculated class values of the measured objects or experiments and their original ones (i.e., the underlying classification hypothesis and the real world). In detail, the following aspects have to be considered when evaluating a solution candidate:

- Calculating the sum of squared errors J between N original class values o_i and calculated class values c_i is a good measurement of the quality of the formula at hand:

$$J = \sum_{i=0}^{N} (o_i - c_i)^2$$

- A good model for the classification schema which is to be identified should classify the test sample objects so that the classes can be separated from each other as well as possible. In other words, each test object's calculated class value should be nearer to the object's original class than to any other class value. This can be measured for a class c as follows:
  Let A be the set of objects that originally belong to the given class c and B the set of all other objects. All members of A are compared to all members of B which means that all pairs in $A \times B = C$ have to be considered. If an element of B has a greater distance to c than an object in A, then this has to be considered as a correct pair; we collect all these correct pairs in D. If classes are perfectly separated from each other, then $D = C$. Thus, the separability S_i of a class c_i is calculated as

$$S_i = \frac{|D_i|}{|C_i|}.$$

  In fact, this value also corresponds to the area under the so-called Receiver Operating Characteristic (ROC) curve. ROC curves have long been used in signal detection theory and are able to visualize the quality of a classification hypothesis; good explanations of ROC curves and how the area under them have to be interpreted are for instance given in [6] and [3].

- The third interesting property of a classification model is how clearly the classes can be separated from each other. This can be measured by calculating the range R_i of the estimated class values for all members of a given class c_i. Of course, the smaller the ranges of the calculated class values of the classes' objects are, the more precise the model seems to be and the better the corresponding GP solution candidate has to be evaluated. In Figure 3 a classification example showing original and estimated class values as well as the resulting class value ranges of the given classes 0, 1, 2 and 3.

At the moment we are trying to find out, how J, S and R have to be weighted (i.e. multiplied with their respective weighting factors j, s and r) in order to maximize the quality of the results produced by the GP algorithm. At the moment these parameters have to be given by the user; recent test studies have shown that it seems to be a good idea to weight the sum of the squared errors J about twice as much as S and R together.

4 Test Results and Discussion of the Operators Used

Empirical studies with various problem instances are possibly the most effective way to analyze the potential of heuristic optimization searches like evolutionary algorithms. Amongst other benchmark problem instances, the GP-based multiclass classification method was tested with the *Thyroid* data set, a benchmark data set which is part of the UCI Machine Learning Repository.

4.1 Results Obtained for the "Thyroid" Benchmark Data Set

This data set represents medical measurements which were recorded while investigating patients potentially suffering from hypotiroidism. A detailed description of the problem can be found on the KEEL homepage. In short, the task is to determine whether a patient is hypothyroid or not. Three classes are formed: normal (not hypothyroid), hyperfunction and subnormal functioning; a good classifier has to be significantly better than 92% simply because 92 percent of the patients are not hyperthyroid.

In total, the data set contains 7200 samples. For testing our GP-based classification method we have used 6000 samples as training samples available for the GP algorithm and tested the model which was automatically produced by the algorithm with the remaining 1200 samples. The algorithm which was used for all test series was the SASEGASA [1], a new generic evolutionary algorithm for retarding the unwanted effects of premature convergence by combining processing and self-adaptive steering of selection-pressure.

Table 1. Classifications overview for the best threshold between the classes "0" and "1" of the Thyroid data set.

Evaluation		
	+	-
+	315 (69.38%)	2 (0.44%)
-	6 (1.32%)	131 (28,85%)
Total: 98.24% correct classifications		

Prognosis		
	+	-
+	51 (63.75%)	0 (0.00%)
-	2 (2.50%)	27 (33.75%)
Total: 97.50% correct classifications		

Table 2. Classifications overview for the best threshold between the classes "1" and "2" of the Thyroid data set.

Evaluation		
	+	-
+	5519 (94.12%)	28 (0.48%)
-	0 (0.00%)	317 (5.41%)
Total: 99.52% correct classifications		

Prognosis		
	+	-
+	1108 (94.70%)	11 (0.94%)
-	0 (0.00%)	51 (4.36%)
Total: 99.06% correct classifications		

This data set represents medical measurements which were recorded while investigating patients potentially suffering from hypotiroidism. A detailed description of the problem can be found on the KEEL homepage. In short, the task is to determine whether a patient is hypothyroid or not. Three classes are formed: normal (not hypothyroid), hyperfunction and subnormal functioning; a good classifier has to be significantly better than 92% simply because 92 percent of the patients are not hyperthyroid.

In total, the data set contains 7200 samples. For testing our GP-based classification method we have used 6000 samples as training samples available for the

http://www.ics.uci.edu/~mlearn/
http://sci2s.ugr.es/keel-dataset/

http://sci2s.ugr.es/keel-dataset/

GP algorithm and tested the model which was automatically produced by the algorithm with the remaining 1200 samples. The algorithm which was used for all test series was the SASEGASA [1], a new generic evolutionary algorithm for retarding the unwanted effects of premature convergence by combining processing and self-adaptive steering of selection-pressure.

The Tables 1 and 2 give an overview of the quality of the identified classifier for the *Thyroid* data set; for both training and prognosis, the confusion matrices for the identified thresholds between the classes 0 and 1 (Fig. 1) and 1 and 2 (Fig. 2) are given. Confusion matrices [9] contain information about actual and predicted classifications done by classification systems. In the upper left cell of the table we state the number of correct predictions that an instance is positive, in the lower right cell the number of correct negative classifications is given. Furthermore, we also state the number of incorrect predictions that an instance is positive (upper right cell) and the number of incorrect predictions that an instance is negative (lower left cell). Additionally, the total percentage of correct classifications is given, too.

The classes could be separated with an average correctness of 98.88% when applied to the training samples and with an average correctness of 98.28% when applied to the remaining test samples.

The interested reader can find further analysis of the results found for this benchmark data set (including ROC curves and the areas under them) and also the results we achieved for other benchmark problems in the results section of the HeuristicLab homepage.

4.2 Operators and Parameter Settings

During our thorough test series we have identified the following GP-relevant parameter settings as the best ones for solving classification problem instances:

- **GP-algorithm:** The SASEGASA. Especially self-adaptive concepts for steering the selection pressure within the population of a genetic algorithm, as they are used by the SASEGASA [1], have successfully been used in the context of model identification for classification data sets. In general, for all multiclass classification problems we investigated the results produced by the SASEGASA are much better than those produced by standard GA implementations.

- **Mutation rate:** 4%.

- **Population size:** 500 - 2,000.

www.heuristiclab.com/results

- **Selection operators:** Whereas standard GA implementations use only one selection operator, the SASEGASA requires two (the so-called female selection operator as well as the male selection operator; for a detailed explanation of this Gender-Specific Selection see [14]). Similar to our experience collected during the tests on the identification of mechatronical systems [15, 16], it seems to be the best to choose the roulette-wheel selection in combination with the random selection operator. The reason for this is that apparently merging the genetic information of rather good individuals (formulae) with randomly chosen ones is the best strategy when using the SASEGASA for solving identification problems.

- **Success ratio, selection pressure and comparison factor:** As for instance described in detail in [1], there are some additional parameters of the SASEGASA regarding the selection of those individuals that are accepted to be a part of the next generation's population. These are the success ratio, the maximal selection pressure and the comparison factor bounds that steer the algorithm's behavior regarding Offspring Selection [2]. For model structure identification tasks in general and especially in case of dealing with classification problems, the following parameter settings seem to be the best ones:

  - Success ratio = 1.0,
  - Maximum selection pressure = 1,000, and
  - Comparison factor = 1.0.

  These settings have the effect that in each generation only offspring survive that are really better than their parent individuals (because of success ratio = 1.0 only better children are inserted into the next generation's population, and because of the maximally high comparison factor a successful offspring really has to be better than both of its parent individuals). This is why the selection pressure becomes very high as the algorithm is executed, and therefore the maximum selection pressure has to be set to a rather high value (as, e.g., 1,000) to avoid premature termination.

- **Crossover operators:** We have implemented and tested three different single-point crossover procedures for GP-based model structure identification: one that exchanges rather big subtrees, one that is designed to exchange rather small structural parts (e.g., only one or two nodes) and one that replaces randomly chosen parts of the

respective structure trees. Moreover, for each crossover operator we have also implemented an extended version that additionally randomly mutates all terminal nodes (i.e., manipulates the parameters of the represented formula). That is, the following 6 structure identification crossover methods are available: *StandardSPHigh*, *StandardSPMedium*, *StandardSPLow*, *ExtendedSPHigh*, *ExtendedSPMedium*, and *ExtendedSPLow*.
Since arbitrarily many crossover operators can be selected when applying the SASEGASA, the task was not to find out which operator can be used to produce the best results but rather which subset of operators is to be chosen. According to what we experienced, the following set of crossover operators should be applied: All three standard operators (*StandardSPHigh*, *StandardSPMedium* and *StandardSPLow*) plus one of the extended ones, for instance *ExtendedSPMedium*.

- **Mutation operators:** The basic mutation operator for GP structure identification we have implemented and tested, *GAStandard*, works as already described in Section 3: A function symbol could become another function symbol or be deleted, the value of a constant node or the index of a variable could be modified. Furthermore, we have also implemented an extended version (*GAExtended*) that additionally randomly mutates all terminal nodes (in analogy to the extended crossover operators).
As the latest test series have shown, the choice of the crossover operators influences the decision which mutation operator to apply to the SASEGASA: If one of the extended crossover operators is selected, it seems to be the best to choose the standard mutation operator. But if only standard crossover methods are selected, picking the extended mutation method yields the best results.

- **Evaluation operators:** As already mentioned in Section 3.2, the sum of squared errors function is widely used for measuring the quality of a classification model; we have also used it as evaluation function during the latest test series. Still, the authors believe that the enhanced evaluation model described in Section 3.2 will help the GP process produce even better classification models; recent tests using prototypical implementations of this evaluation scheme have yielded very promising results.

Selected experimental results of the standard GP implementation and the SASEGASA algorithm for the *Thyroid* data set using various parameter settings are presented in Table 3. For each parameter settings version 5 independent test runs were executed, the best results are listed. In all cases, the population size was 1000 and the mutation rate set to 4%. Furthermore, the following parameter settings were used:

Table 3. Experimental results for the "Thyroid" data set.

Using standard GP implementation		
Parameter settings	*Correct classifications*	
	Evaluation	*Prognosis*
(1)	92.80%	92.13%
(2)	93.91%	93.25%

Using the SASEGASA		
Parameter settings	*Correct classifications*	
	Evaluation	*Prognosis*
(3)	97.15%	96.34%
(4)	98.21%	98.07%
(5)	97.91%	97.42%
(6)	97.70%	97.25%
(7)	**98.88%**	**98.28%**

- **(1)**: crossover: *ExtendedSPMedium*; mutation: *GAStandard*; selection: *Roulette*.
- **(2)**: crossover: *StandardSPMedium*; mutation: *GAExtended*; selection: *Roulette*.
- **(3)**: crossover: *all 6 available operators*; mutation: *GAExtended*; selection: *Random* and *Roulette* (max. selection pressure: 500, comparison factor bounds: 0.6 to 0.9).
- **(4)**: crossover: *all 6 available operators*; mutation: *GAStandard*; selection: *Random* and *Roulette* (max. sel. pressure: 500, comparison factor: 1.0).
- **(5)**: crossover: *all 3 standard operators plus ExtendedSPLow*; mutation: *GA-Standard*; selection: *Random* and *Roulette* (max. selection pressure: 500, comparison factor bounds: 0.6 to 0.9).
- **(6)**: crossover: *all 3 standard operators plus ExtendedSPLow*; mutation: *GA-Standard*; selection: *Roulette* and *Roulette* (max. selection pressure: 500, comparison factor: 1.0).
- **(7)**: crossover: *all 3 standard operators plus ExtendedSPLow*; mutation: *GA-Standard*; selection: *Random* and *Roulette* (maximum selection pressure: 500, comparison factor: 1.0).

In comparison to previously published results obtained for the *Thyroid* data set using other approaches

Table 4. Summary of the best GP parameter settings for solving classification problems.

GP algorithm	SASEGASA
Mutation Rate	4%
Population Size	1,000
Selection Operators	Random, Roulette
Maximum Selection Pressure	1,000
Comparison Factor	1.0
Crossover Operators	StandardSPLow, StandardSPMedium, StandardSPHigh, ExtendedSPLow
Mutation Operator	GAStandard

for solving classification problems, these results seem to be remarkably good. E.g., in [5] a classification method based on Neural Networks is presented yielding a *Thyroid* classificator that incorrectly classifies 5.67% of the validation samples and 6.67% of the test samples. In [7], for instance, several other classification methods are discussed, the best one producing results with a misclassification rate of 5.4%. The best parameter settings for solving classification problems using the presented GP-based method are subsumed in Table 4.

5 Conclusion

In this paper we have introduced a genetic programming based method that is able to solve nontrivial real-world multiclass classification problems. A GP approach for identifying nonlinear model structures for mechatronical systems has been further developed and refined so that the basic methods can be used for attacking classification problems yielding very satisfying results. Especially, new hybrid variants of genetic algorithms supplementing standard genetic algorithms with artificial self organizing aspects for overcoming some of the fundamental problems of GAs and GP have been applied quite successfully. On the basis of recent test results the authors strongly believe that further refinement of the available operators as well as the design of new operators for this kind of optimization problems will make it possible to find even better results in the field of data mining and classification.

References

[1] M. Affenzeller and S. Wagner. SASEGASA: A new generic parallel evolutionary algorithm for achieving highest quality results. *Journal of Heuristics - Special Issue on New Advances on Parallel Meta-Heuristics for Complex Problems*, 10:239–263, 2004.

[2] M. Affenzeller and S. Wagner. Offspring selection: A new self-adaptive selection scheme for genetic algorithms. *Adaptive and Natural Computing Algorithms*, pages 218–221, 2005.

[3] A. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30:1145–1159, 1997.

[4] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery: An overview. *Advances in Knowledge Discovery and Data Mining*, 1996.

[5] F. Hamker and D. Heinke. Implementation and comparison of growing neural gas, growing cell structures and fuzzy artmap. Technical Report ISSN 0945-7518, Technische Universität Ilmenau, 1997.

[6] J. Hanley and B. McNeil. The mening and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143:29–36, 1982.

[7] S. Hochreiter and K. Obermayer. Classification, regression, and feature selection on matrix data. Technical Report ISSN 1436-9915, Department of Electrical Engineering and Computer Science, Technische Universitat Berlin, 2004.

[8] J. H. Holland. *Adaption in Natural and Artificial Systems, 1st MIT Press ed.* 2004.

[9] R. Kohavi and F. Provost. Glossary of terms. *Machine Learning - Special Issue on Applications of Machine Learning and the Knowledge Discovery Process*, 30:271 – 274, 1998.

[10] J. Koza. *Genetic Programming: On the Programming of Computers by means of Natural Selection.* The MIT Press, Cambridge, Mass, 1992.

[11] W. Langdon and R. Poli. *Foundations of Genetic Programming.* Springer Verlag, Berlin Heidelberg New York, 2002.

[12] T. M. Mitchell. *Machine Learning.* McGraw-Hill, New York, 2000.

[13] S. Wagner and M. Affenzeller. Heuristiclab: A generic and extensible optimization environment. *Adaptive and Natural Computing Algorithms*, pages 538–541, 2005.

[14] S. Wagner and M. Affenzeller. Sexual GA: Gender-specific selection for genetic algorithms. *Proceedings of the 9th World Multi-Conference on Systemics, Cybernetics and Informatics (WMSCI)*, pages 76–81, 2005.

[15] S. Winkler. Identification of nonlinear model structures by genetic programming techniques. Master's thesis, Institut für Systemtheorie und Simulation, Technisch-Naturwissenschaftliche Fakultät der Johannes Kepler Universität, Linz, Austria, 2004.

[16] S. Winkler, M. Affenzeller, and S. Wagner. New methods for the identification of nonlinear model structures based upon genetic programming techniques. *Journal of Systems Science*, 31(1):5–13, 2005.