2

Evolutionary Systems Identification: New Algorithmic Concepts and Applications

Michael Affenzeller, Stephan Winkler and Stefan Wagner
Upper Austrian University of Applied Sciences
School of Informatics, Communications and Media
Heuristic and Evolutionary Algorithms Laboratory
Softwarepark 11, 4232 Hagenberg,
Austria

1. Introduction

There are many problems in various theoretical and practical disciplines that require robust structure identification techniques with as few restricting assumptions in terms of model structure and potentially relevant input variables (features) as possible. Due to its implicit variable selection and the possibility to identify also nonlinear model structures, the basic concept of Genetic Programming (GP) has the required descriptive potential and provides results in form of easily interpretable formulae as an additional benefit. However, when using standard GP techniques, the potential of GP is still rather limited and restricted to special applications.
This chapter presents further developed algorithmic concepts which can be combined with a Genetic Algorithm (GA) as well as with Genetic Programming (GP). Especially the latter combination provides a very powerful, generic and stable algorithm for the identification of nonlinear systems, no matter if the application at hand is in the context of regression, classification or time-series analysis.
After a general introduction in heuristic optimization and Evolutionary Algorithms, the further developed algorithmic concepts are explained. Furthermore, some exemplary applications of Genetic Programming to data based system identification problems are illustrated.

2. Heuristic optimization

Many practical and theoretical optimization problems are characterized by their highly multimodal search spaces. These problems include NP-hard problems of combinatorial optimization, the identification of complex structures, or multimodal function optimization. In the area of production planning and logistics such problems occur especially frequently (as for example task allocation, routing, machine sequencing, container charging). The application of conventional methods of Operations Research (OR) like dynamic programming, the simplex method, or gradient techniques, often fails for these kinds of problems, because the computation effort grows exponentially with the problem dimension.
Therefore, heuristic methods with much lower computational costs are applied quite frequently, even if they can no longer assure the achievement of a global optimal solution.

Source: Advances in Evolutionary Algorithms, Book edited by: Witold Kosiński, ISBN 978-953-7619-11-4, pp. 468, November 2008, I-Tech Education and Publishing, Vienna, Austria

About three decades ago, inspired by nature, literature started to discuss generic heuristic methods which often surpass problem specific heuristics and are moreover much more flexible concerning modifications in the problem definition.
These optimization techniques derived from nature include Simulated Annealing (SA) which draws an analogy between the annealing of material to its lowest energetic state and an optimization problem, or Evolutionary Algorithms (EAs) which are basically inspired by biological evolution. Further recent approaches like Tabu Search (TS), Ant-Colony Optimization (ACO), or Particle Swarm Optimization (PSO) are also mentionable in the context of bionically inspired optimization techniques. Agent theory is also on the verge of achieving greater importance in the field of heuristic optimization.

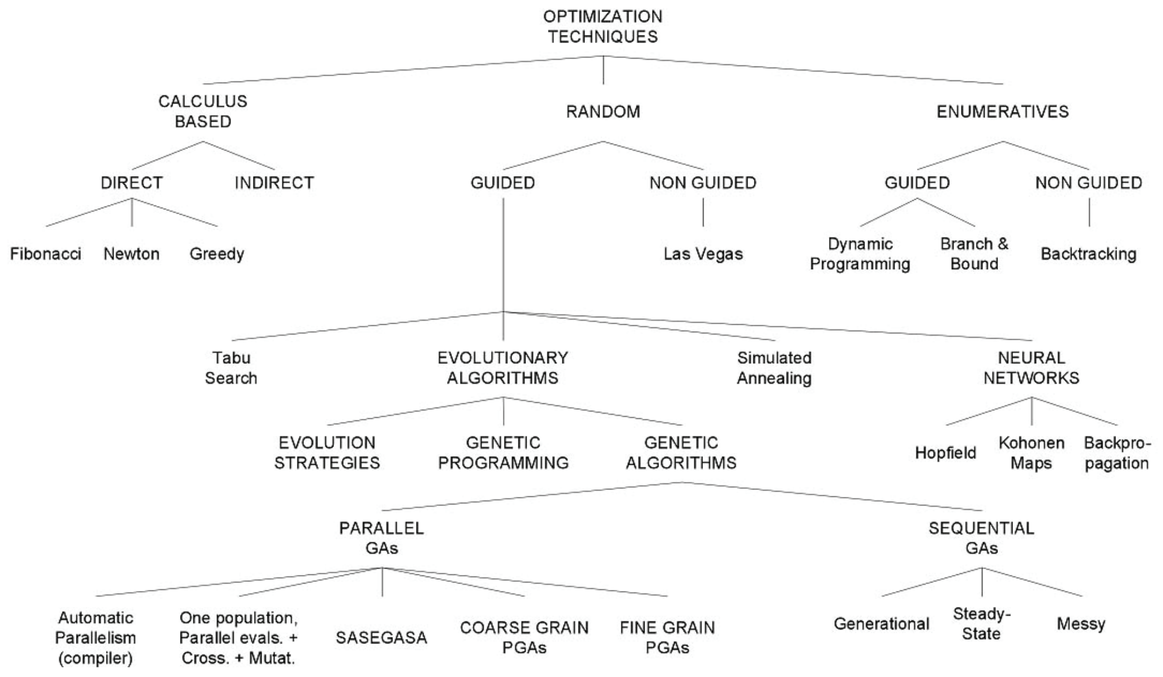


Fig. 1. Taxonomy of optimization techniques

A well-established taxonomy of optimization techniques is given in Fig. 1 whereby our classification describes those classes of methodologies in more detail which are more relevant in the context of the present contribution. This has especially been done for the class of Evolutionary Algorithms which is described in further detail in the following section. The detailed analysis of variants of Genetic Algorithms as shown in Fig. 1 can in principle also be applied to Genetic Programming since it is based on the same algorithmic and methodological concepts.

3. Evolutionary computation

3.1 Evolutionary algorithms: genetic algorithms, evolution strategies and genetic programming

Literature generally distinguishes Evolutionary Algorithms into Genetic Algorithms (GAs), Evolution Strategies (ES), and Genetic Programming (GP).
Genetic Algorithms, possibly the most prevalent representative of Evolutionary Computation, were first presented by Holland (Holland, 1975). Based upon Holland's ideas the concept of the Standard Genetic Algorithm (SGA), which is still very much influenced by the biological archetype, became accepted (described e.g. in (Tomassini, 1995). Due to the

enormous increase of computational power since 1975, the potential of GAs has been tapped more and more. Consequently the popularity of GA-concepts increased steadily and many groups around the world started to solve various problems with GAs. However, it soon became clear that for most practical tasks the binary encoding originally used by Holland was not at all sufficient. Accordingly many different encodings, and also necessary new crossover and mutation operators, were introduced which showed qualitatively very diverse behavior. An overview of different encodings and operators developed for various applications can for instance be found in (Dumitrescu et al., 2000). Since then GAs have been successfully applied to a wide range of problems including many combinatorial optimization problems, multimodal function optimization, machine learning, and the evolution of complex structures such as neural networks. An overview of GAs and their implementation in various fields is given by Goldberg (Goldberg, 1989) and Michalewicz (Michalewicz, 1996).
Evolution Strategies, the second major representative of Evolutionary Algorithms, were introduced by Rechenberg (Rechenberg, 1973) and Schwefel (Schwefel, 1994). Evolution Strategies tend to find local optima quite efficiently. Though, in the case of multimodal solution spaces, Evolution Strategies tend to detect a global optimum hardly, if none of the starting values is located in the absorbing region of such a global optimum. Nevertheless, ES have lead the way in the implementation of self-adaptive concepts in the area of Evolutionary Computation and are considered one of the most powerful and efficient concepts for the optimization of real-valued parameter vectors.
Genetic Programming (GP) has been established as an independent branch in the field of Evolutionary Computation even if this technique could also be interpreted as a special class of GAs. Based on the basic considerations of Koza (Koza, 1992) to interpret the underlying problem representation in a more general and dynamic way than a usual GA, the basic mechanisms of selection, recombination, and mutation are adapted and applied in a similar manner as found within GAs. The more general problem representation of GP allows the definition of individuals of a population as structures, formulas, or even more generally as programs. This allows the consideration of new applications of EAs like data based systems identification, for example; however, it still seems to be a very ambitious goal to generate more complex programs by means of Genetic Programming.

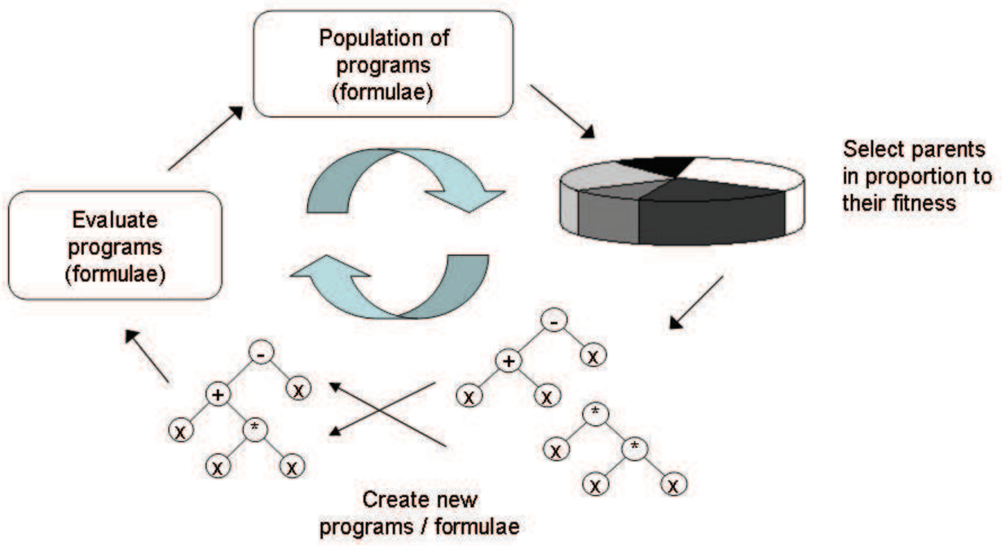

Population of programs (formulae)
Select parents in proportion to their fitness
Evaluate programs (formulae)
Create new programs / formulae

Fig. 2. The GP Lifecycle (Langdon & Poli, 2002)

In (Koza, 1992) it has been 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: A population of solution candidates evolves through many generations towards a solution using evolutionary operators (crossover and mutation) and a "survival-of-the-fittest" selection scheme. 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 solving the optimization problem at hand. 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 (Langdon & Poli, 2002). This procedure is graphically illustrated in Fig. 2.

3.2 Considerations about selected theoretical aspects of evolutionary computation techniques

Fig. 1 indicates that this classification - especially of the bionic methods - is mainly inspired by the natural role-model. For a more directed consideration of algorithmic concepts of the different methods, it is reasonable to differentiate these methods by their basic idea. One possible (and especially in the context of further considerations drawn in this paper) well-suited classification is the distinction between neighbourhood-based and non-neighbourhood-based search techniques as illustrated in Fig. 3.

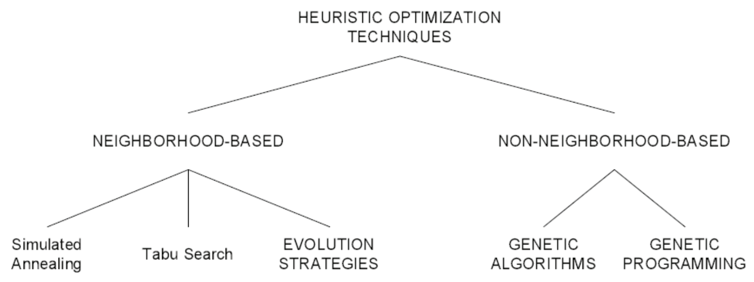


Fig. 3. Classification of heuristic optimization techniques due to their mode of operation

As some kind of approximation for the gradient information which is not available for problems of combinatorial optimization, a conventional neighbourhood search aims to obtain information about the descent/increase of the objective function in the local neighbourhood at a certain point. Conventional neighbourhood searches start from an arbitrary point in the solution space and iteratively move to more and more promising points along a given neighbourhood structure (with respect to the objective function) as long as no better solution can be detected in the local neighbourhood. The self-evident drawback of this method is that for more complex functions the algorithm converges and gets stuck in the next attracting local optimum which is often far away of a global optimum. It is a common feature of all methods based upon neighbourhood searches to counteract this essential handicap. Simulated Annealing, on the one hand, also allows moves to worse neighbourhood solutions with a certain probability which decreases as the search process progresses in order to scan the solution space broader at the beginning, and to become more and more goal-oriented as the search process goes on. A Tabu Search on the other hand introduces some kind of memory in terms of a so-called tabu list which stores moves that are considered to lead to already visited areas of the search space. However, Evolution Strategies (ES), a well-known representative of Evolutionary Computation, also have to be considered as some kind of parallel neighbourhood search, as asexual mutation (a local operator) is the only way to create new individuals (solution candidates) in the standard ES-versions. Therefore, in the case of multimodal test functions, global optima can be detected by Evolution Strategies only if one of the starting values is located in the absorbing region of a global optimum.

Genetic Algorithms (and certainly also GP), the non-neighbourhood-based search techniques in our classification of heuristic methods, take a fundamentally different approach to optimization, by considering recombination (crossover) as their main operator, whereas the essential difference to neighbourhood-based techniques is given by the fact that recombination is a sexual operator, i.e. properties of individuals from different regions of the search space are combined in new individuals. Therefore, provided that the used problem representation and the operators are adequate, the advantage of applying GAs to hard optimization problems lies in their ability to search broader regions of the solution space than heuristic methods based upon neighbourhood search do. Nevertheless, GAs are also frequently faced with a problem which, at least in its impact, is quite similar to the problem of stagnating in a local but not global optimum. This drawback, called premature convergence in the terminology of GAs, occurs when the population of a GA reaches such suboptimal state that the genetic operators can no longer produce offspring which outperform their parents (Fogel, 1994).

A very essential question about the general performance of a GA is, whether or not good parents are able to produce children of comparable or even better fitness (the building block hypothesis implicitly relies on this). In natural evolution, this is almost always true. For Genetic Algorithms this property is not so easy to guarantee. The disillusioning fact is that the user has to take care of an appropriate coding in order to make this fundamental property hold. In order to overcome this strong requirement we have developed an advanced selection mechanism (Affenzeller & Wagner 2004) which is based on the idea to consider not only the fitness of the parents, in order to produce a child for the ongoing evolutionary process. Additionally, the fitness value of the evenly produced offspring is compared with the fitness values of its own parents. The offspring is accepted as a candidate for the further evolutionary process if and only if the reproduction operator was able to produce an offspring that could outperform the fitness of its own parents. This strategy guarantees that evolution is presumed mainly with crossover results that were able to mix the properties of their parents in an advantageous way. Via these means we are already in a position to attack one of the reasons for premature convergence. Furthermore, this strategy has proven to act as a precise mechanism for self-adaptive selection pressure steering, which is of major importance in the migration phases of parallel Evolutionary Algorithms. All these new generic concepts are very promisingly combined in the SASEGASA-algorithm (Affenzeller & Wagner, 2004). Even if the aspect of parallelization is mainly used to improve global convergence in our research so far, the next obvious step is to transform these massively parallel concepts to parallel computing environments. Furthermore, already established parallel GAs should benefit from the recently developed new theoretical concepts as the essential genetic information can be assembled much more precisely in the migration phases.

4. Advanced algorithmic concepts for genetic algorithms

4.1 General remarks on variable selection pressure within genetic algorithms

Our first attempt for adjustable selection pressure handling was the so-called Segregative Genetic Algorithm (SEGA) (Affenzeller, 2001) which introduces birth surplus in the sense of a (μ, λ)-Evolution Strategy (Beyer, 1998) into the general concept of a GA and uses this enhanced flexibility primary for adaptive selection pressure steering in the migration phases of the parallel GA in order to improve achievable global solution quality. The SASEGASA, which stands for Self Adaptive Segregative Genetic Algorithm with aspects of Simulated Annealing, is a further development of SEGA and distinguishes itself mainly in its ability to self-adaptively adjust selection pressure in order to achieve progress in solution quality without loosing essential genetic information which would lead to unwanted premature convergence. The SASEGASA is generic in that sense that all algorithmic extensions are problem-independent so that they do not depend on a certain problem representation and the corresponding operators.

Therefore we have decided to combine the further deloped algorithmic concepts of SASEGASA with Genetic Programming (GP). However, we have observed two major differences when combining SASEGASA and Genetic Programming compared to the experience in the application of SASEGASA in other domains like combinatorial optimization or real-valued optimization (Affenzeller, 2005):

- The potential in terms of achievable solution quality in comparison with the standard algorithms seems to be considerably higher in the field of GP than in standard applications of GAs.
- By far not all algorithmic extensions of SASEGASA are relevant in GP. Only some algorithmic aspects of the rather complex SASEGASA concept are really relevant in the GP domain which makes the handling and especially parameter adjustment easier and more robust.

Therefore, the discussion in this article will focus on the algorithmic parts of SASEGASA which are really relevant for GP. In doing so, this section is structured as follows: The first subsection describes the general idea of SASEGASA in a quite compact way, whereas the second subsection focusses on that parts of SASEGASA in further detail which are really relevant for the present contribution and discusses the reasons for that.

For a more detailed description of all involved algorithmic aspects the interested reader is referred to the book (Affenzeller, 2005).

In principle, the SASEGASA introduces two enhancements to the basic concept of Genetic Algorithms. Firstly, it brings in a variable and self-adaptive selection pressure in order to control the diversity of the evolving population in a goal-oriented way w.r.t. the objective function. The second concept introduces a separation of the population to increase the broadness of the search process and joins the subpopulation after their evolution in order to end up with a population including all genetic information sufficient for locating a global optimum.

At the beginning of the evolutionary process the whole population is divided into a certain number of subpopulations. These subpopulations evolve independently from each other until the fitness increase stagnates in all subpopulations because of too similar individuals within the subpopulations, i.e. local premature convergence. Thanks to offspring selection this can be triggered exactly when an upper limit of selection pressure is exceeded (cf. Subsection 4.2). Then a reunification from *n* to *(n-1)* subpopulations is performed by joining an appropriate number of adjacent subpopulation members.

Metaphorically speaking this means, that the villages (subpopulations) at the beginning of the evolutionary process are slowly growing together to bigger towns, ending up with one big city containing the whole population at the end of evolution. By this approach of width-search essential building blocks can evolve independently in different regions of the search space at the beginning and during the evolutionary process.

4.2 Offspring selection in SASEGASA

In (Affenzeller & Wagner, 2004) it has been shown that the aspect of segregation and reunification is highly relevant in order to systematically improve the achievable global solution quality of combinatorial optimization problems as for example the travelling salesman problem (TSP). Still, we have not used this parallel approach for our GP-based modelling studies. On the one hand, this would lead to a high increase of runtime consumption; on the other hand, anyway, we do not expect any significant increase of solution quality using this concept for GP-based modelling as results summarized in (Affenzeller, 2005) indicate that this parallel approach does not remarkably effect the solution quality of optimization problems others than combinatorial problems.

A very essential question about the general performance of GAs or GP is, whether or not good parents are able to produce children of comparable or even better fitness (the building block hypothesis implicitly relies on this). In natural evolution, this is almost always true. For artificial evolution and exceptionally for Genetic Programming this property is not so easy to guarantee. Offspring selection assures exactly that property.

Offspring selection considers not only the fitness of the parents, in order to produce a child for the ongoing evolutionary process. Additionally, the fitness value of the evenly produced offspring is compared with the fitness values of its own parents. The offspring is accepted as a candidate for the further evolutionary process if and only if the reproduction operator was able to produce an offspring that could outperform the fitness of its own parents. This strategy guarantees that evolution is presumed mainly with crossover results that were able to mix the properties of their parents in an advantageous way.

As in the case of conventional GAs, or GP, offspring are generated by parent selection, crossover, and mutation. In a second (offspring) selection step, the number of offspring to be generated is defined to depend on a predefined ratio-parameter giving the quotient of next generation members that have to outperform their own(!) parents (success ratio, *SuccRatio*). As long as this ratio is not fulfilled, further children are created and only the successful offspring will definitely become members of the next generation; this procedure is illustrated in Fig. 4. When the postulated ratio is reached, the rest of the next generation members are randomly chosen from the children that did not reach the success criterion. Within our new selection model, selection pressure is defined as the ratio of generated candidates to the population size. An upper limit for selection pressure gives a quite intuitive termination heuristics: If it is no more possible to find a sufficient number of offspring that outperform their parents, the algorithm terminates in the simple version as being used here or new genetic information is brought in by reunification in the more general formulation of the parallel SASEGASA.

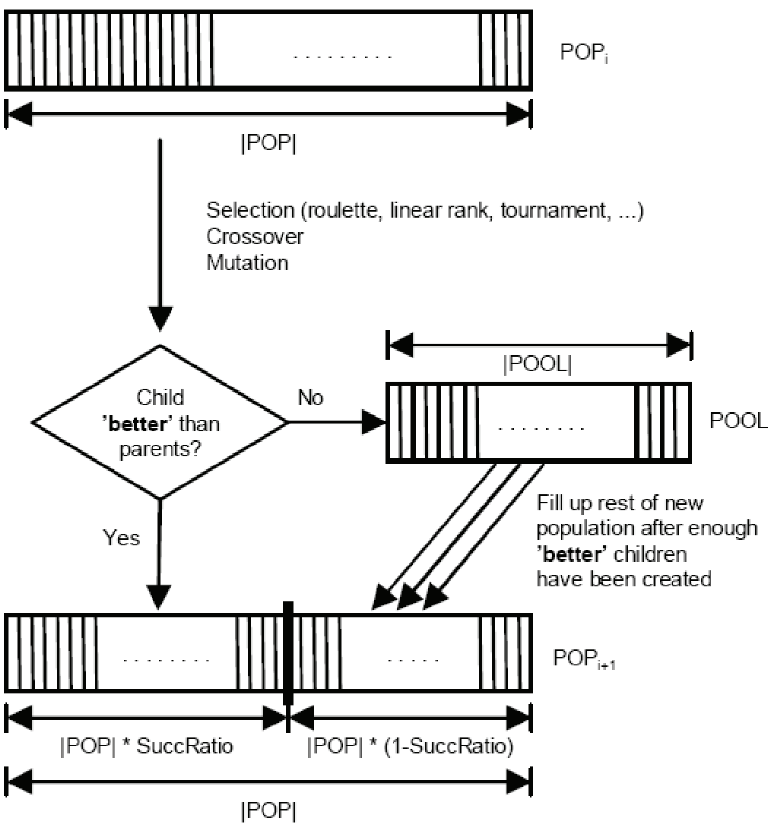

POP$_i$
|POP|
Selection (roulette, linear rank, tournament, ...)
Crossover
Mutation
|POOL|
Child 'better' than parents?
No
POOL
Yes
Fill up rest of new population after enough 'better' children have been created
POP$_{i+1}$
|POP| * SuccRatio
|POP| * (1-SuccRatio)
|POP|

Fig. 4. Flowchart for embedding offspring selection into a Genetic Algorithm.

5. Data based systems identification

Data mining is understood as the practice of automatically searching large stores of data for patterns. Incredibly large (and quickly growing) amounts of data are collected not only in commercial, administrative, and scientific, but also in medical databases; this is the reason why intelligent computer systems that can extract useful information (such as general rules or interesting patterns) from large amounts of observations are needed. In short, "data mining is the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data" (Fayyad et al. 1996). This is why data based machine learning algorithms have to be applied in order to retrieve additional insights into human biological processes, how environment factors influence human health or how certain human parameters are related. The following three classes of data analysis problems are relevant within medical data analysis: Regression, classification and time series analysis. In any of these cases, statistical algorithms are supposed to "learn" functions by analyzing a set of input-output examples ("training samples").

In statistics, **regression analysis** is understood as the act of modelling the relationship between variables, namely between one or more target ("dependent") variables and other variables (also called input or explanatory variables). I.e., the goal is to find a mathematical function f which can be used for calculating the target variable Y using the input variables $X_{1..p}$:

$$Y = f(X_1, ..., X_p)$$

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. A statistical classification algorithm is supposed to take feature representations of objects and map them to a special, predefined classification label. Such a classification algorithm is designed to learn a function f which maps a vector of object features $X_1,\ldots,X_p$ into one of several classes. A given sample x_i can so be classified using f and $X_1,\ldots,X_p$:

$$\text{Class}(x_i) = f(X_{1(i)}, \ldots, X_{p(i)})$$

There are several approaches which are nowadays used for solving classification problems; the most common ones are (as described in (Mitchell, 2000), e.g.) decision tree learning, instance-based learning, inductive logic programming (such as in Prolog, e.g.) and reinforcement learning.

Finally, there are two main goals of **time series analysis**: On the one hand one tries to identify the cause of a phenomenon represented by a sequence of observations and its relationships with other sequences of observations, and on the other hand the goal is to predicting future values of time series variables. Both of these goals require that the pattern of observed time series data is identified and more or less formally described. I.e., for the target variable Y one wants to identify a function f so that Y at time t can be calculated using values of other variables and (if available) also information about the history of Y:

$$Y(t) = f(X_{1(t-\{0..z\})}, \ldots\ldots, X_{p(t-\{0..z\})}, Y_{(t-\{0..z\})})$$

where z is the maximum time offset for variables used in f. Detailed discussions of time series and methods applicable can for example be found in (Box & Jenkins, 1976) or Kendall & Ord, 1990).

6. GP-Based structure identification

6.1 Introduction, general remarks

The concept of structure identification is not very common in the literature. Indeed, it is well known that every model consists of an equation set (the structure) and of values (parameters). System identification actually implies both, but usually the definition of the structure is considered either obvious or as the less critical issue, while the consistent estimation of the parameters especially in presence of noise receives the largest part of the attention. By its very general problem statement, GP allows to approach the problem of structure identification and the problem of parameter identification simultaneously. As a consequence, GP techniques are used for identifying various kinds of technical systems; some approaches use genetic programming to identify the structure in addition to standard parameter estimation techniques, many other ones use GP for determining both the structure and the parameters of the model of a nonlinear system as for example described in (Rodriguez et al., 2000) and (Beligiannis et al., 2005).

GP-based, data driven systems identification 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 system's target values. 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 that maps a vector of input values to the respective target values. In other words, each presented instance of the structure identification problem is interpreted as an instance of an optimization problem; a solution is found by a heuristic optimization algorithm. Further details about the operators used are given for example in (Winkler et al., 2006a). The goal of the implemented GP identification process is to produce an algebraic expression from a database containing the measured results of the experiments to be analyzed. Thus, 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:

- *Nonterminal* nodes represent functions performing some actions on one or more property values within the structure to produce the values of the target property (which 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; as with every GP modeling process, the selection of the library functions is an important part since this library should be able to represent a wide range of systems. When the evolutionary algorithm is executed, each individual of the population represents one structure tree.

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 but it can help the optimization algorithm to reintroduce genetic diversity in order to re-stimulate genetic search.

Examples of genetic operations on tree structures are shown in Fig. 5: The crossover of parent1 (representing the expression "$5/x_1(t-5)+\ln(x_2(t-2))$" and parent2 ("$x_3(t) * x_2(t-1)-1.5$") yields child1 ("$5/x_1(t-5)+x_3(t)*x_2(t-1)$"), child2 and child3 are possible mutants of child1 representing "$5/x_1(t-5)+x_3(t)$" and "$5-x_1(t-5)+x_3(t-1)*x_2(t)$".

Since the GP algorithm tries to maximize or minimize some objectiv 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 data based modeling, this function should be an appropriate measure of the level of agreement between the original target variable's values and those calculated using the model to be evaluated. Calculating the sum of squared errors J between original values o_i and calculated values c_i is a simple as well as robust measurement of the quality of the formula at hand:

$$\mathrm{J} = \sum_{i=1}^{n}(o_i - e_i)^2$$

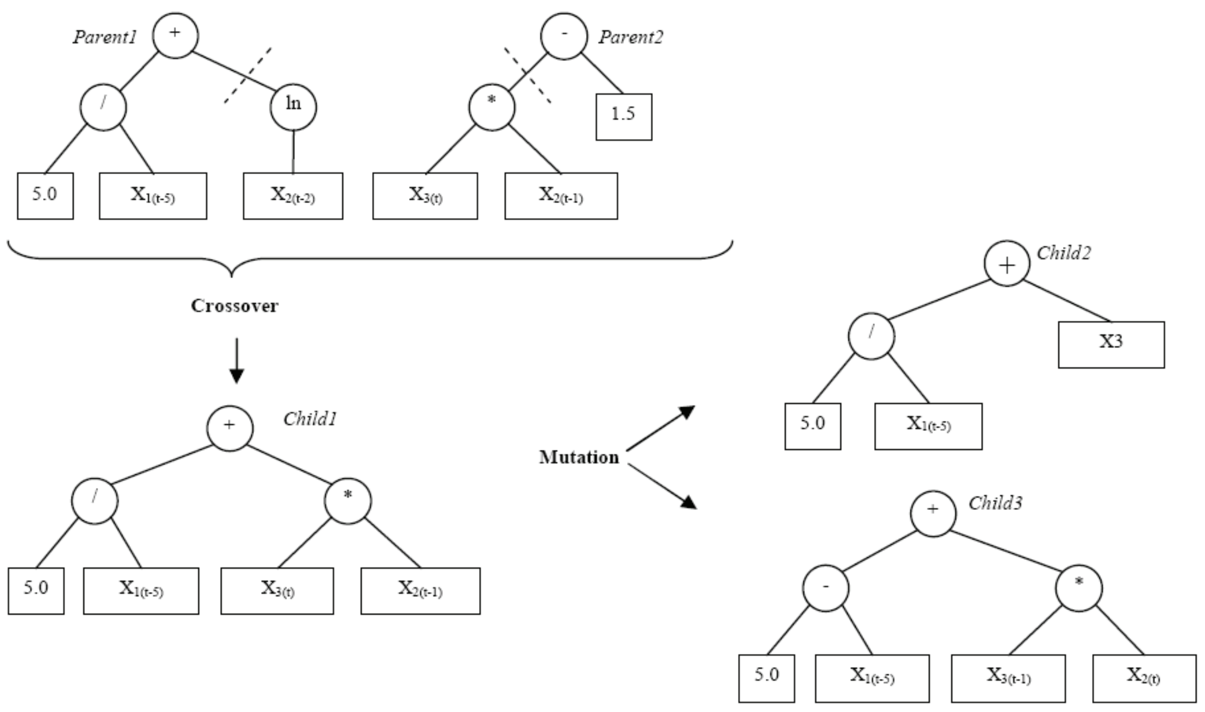


Fig. 5. Genetic Operations on Tree Structures.

6.2 GP Based structure identification: the HeuristicModeler

On the basis of preliminary work on the identification of nonlinear structures in dynamic technical systems (Winkler et al, 2005a), (Winkler et al, 2005b), (Del Re et al, 2005) as well as several other enhanced algorithmic and problem specific mechanisms we have implemented the HeuristicModeler (Winkler et al, 2006c), a multi-purpose machine learning algorithm that is able to evolve models for various different machine learning problem classes. The framework used for the implementation of the HeuristicModeler is the HeuristicLab (Wagner & Affenzeller, 2005), a framework for prototyping and analyzing optimization techniques for which both generic concepts of evolutionary algorithms and many functions for analyzing them are available.

The algorithmic basis for the HeuristicModeler is the SASEGASA (for an explanation see Section 4. There are several new hybrid evolutionary concepts combined in this algorithmic basis, the most important ones being on the one hand the self-adaptive selection pressure steering and on the other hand the so-called Offspring Selection concept.

The selection pressure measures how hard it is to produce individuals out of the current population that improve the overall fitness. As soon as this internal selection pressure reaches a pre-defined maximum value, the algorithm is terminated and presents the best actual model as the result of the training process. Details can be found in (Affenzeller & Wagner, 2004) and (Affenzeller, 2005).

As already explained in further detail in Section 4, the basic idea of Offspring Selection is that individuals are first compared to their own parent solution candidates and accepted as members of the new generation's population if they meet certain criteria. In the context of structure identification and machine learning we have realized that the use of very rigid settings yields best results (Winkler et al., 2006b).

Research results obtained within the last two years have lead to the conclusion that a simplified version of offspring selection together with a slightly modified parent selection shows the best and most robust results in the context of GP-applications. Thus, *SuccRatio* should be set to *1.0*, i.e. every offspring for the next generation is forced to pass the success criterion. Furthermore, it is beneficial in GP applications to state that a child is better than its parents if and only if it is better than the better of the two parents. In the context of combinatorial optimization problems where some intermediate value of the parents fitness values is used as a threshold value for the success criterion, such settings would massively tend to support premature convergence. But in the field of Genetic Programming applications these parameter settings lead to high-quality results quite robustly.

However, there is one aspect concerning parent selection that is to considered in this application domain. It is - applying the parameter settings of offspring selection mentioned above – most effective to use different selection methods for the selection of the two parents which are chosen for crossover. In the present context this gender specific selection aspect (Wagner & Affenzeller, 2005) is implemented most effectively by selecting one parent conventionally by roulette-wheel selection and the other parent randomly.

All together, this especial variant of adapted sexual selection combined with a simplified version of offspring selection aims to cross one above-average parent with a randomly selected parent (which brings in diversity) as long as a whole new population could be filled up with children that were better than their better parent. An upper limit for selection pressure acts as termination criterion in that sense that the algorithm stops, if too many trials (|*POP*| * *maxSelPress*) were already taken and still no new population consisting of successful offspring could be generated. In other words, this indicates that it is not possible to generate a sufficient amount of children that outperform their parents out of the current gene pool; obviously, this seems to be a reasonable termination criterion for an Evolutionary Algorithm. This special version of SASEGASA or offspring selection respectively is schematically shown in Fig. 6.

The GP-based structure identification methods described in the previous section have been implemented as plug-ins for the HeuristicLab forming the problem specific basis of the HeuristicModeler. The following modeling specific extensions have been integrated into the general GP workflow:

- During the execution of a structure identification algorithm it can easily happen that a model showing a very suitable structure is assigned a very bad fitness value only due to inadequate parameter settings. Therefore we have implemented an additional local parameter optimization stage based on real-values encoded Evolution Strategies and integrated it into the execution of the Genetic Programming algorithm.
- As the GP-based model training algorithm tries to evolve better models, it can easily happen that models become more and more complex; the more complex models are, the better they can fit given training data, but they are also negative effects, namely increasing runtime consumption as well as the danger of overfitting. Therefore a heuristic tree pruning algorithm has also been integrated into the HeuristicModeler; in certain intervals, selected models included in the actual models pool are selected and pruned systematically, i.e. formula parts that do not seem to have a measurable influence on the model's evaluation are deleted in order to retrieve simpler models without significantly losing quality.

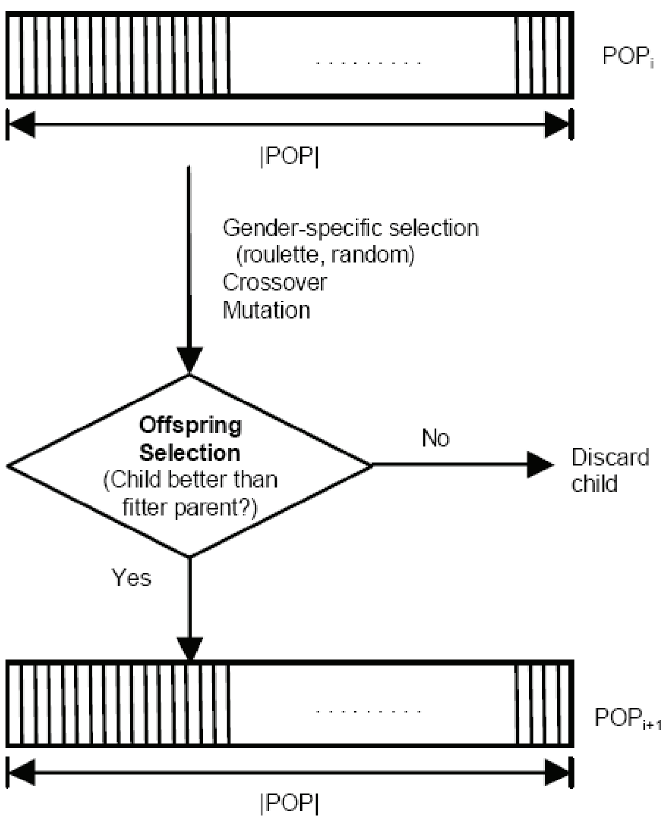


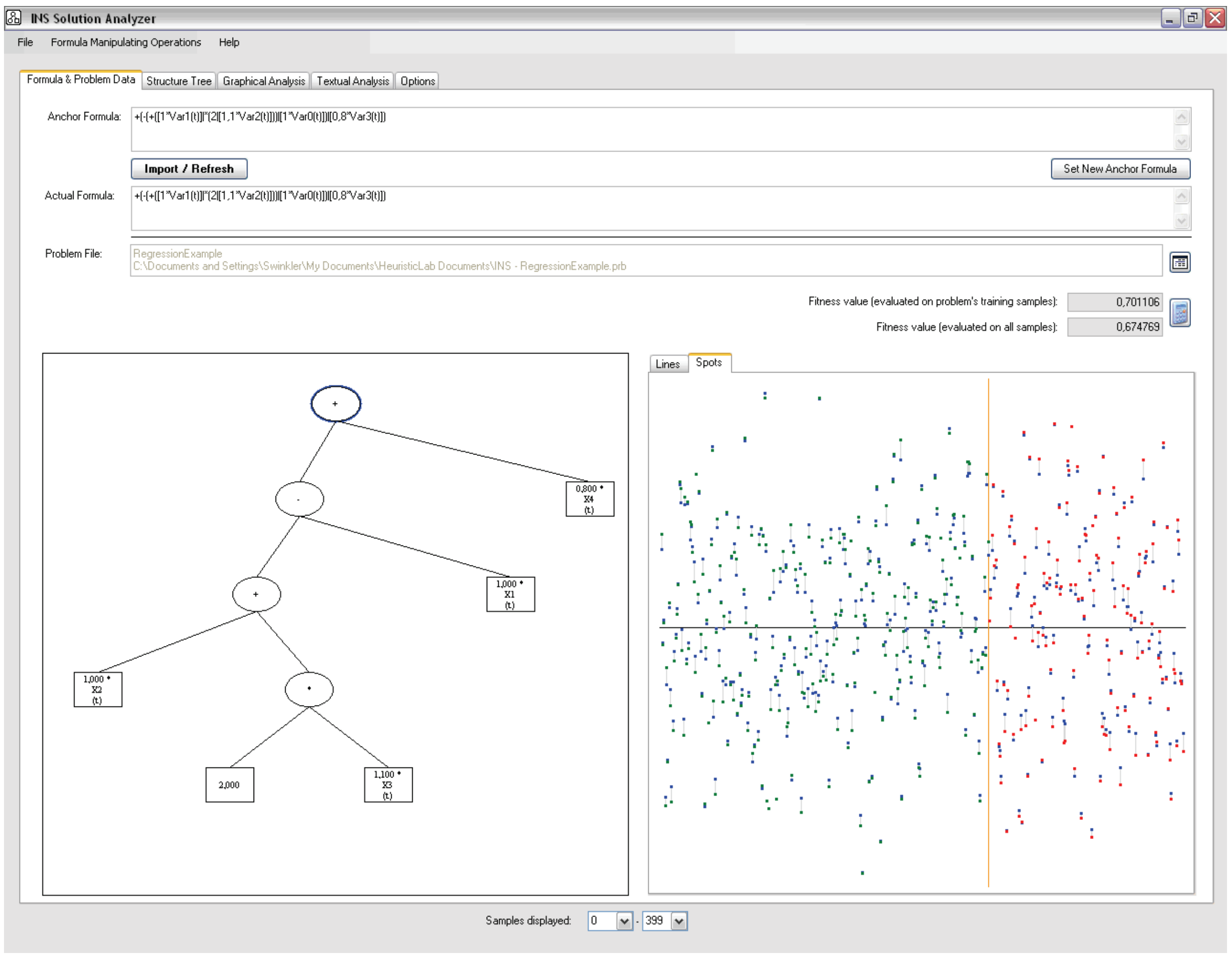
Fig. 6. Flowchart for embedding a simplified version of offspring selection into the GP process.

Due to its flexible and wide functional basis and the extended concepts described above, the GP-based modelling concept implemented in the HeuristicModeler is less exposed to the danger of overfitting than other machine learning algorithms; recent results and comparisons to other data-based modelling techniques are for example summarized in (DelRe et al., 2005), (Winkler et al, 2006f) and (Winkler et al., 2006a). Furthermore, as we will show in the following section, the results generated using the HeuristicModeler can easily be analyzed and interpreted using the HeuristicModelAnalyzer, a tool for analyzing solutions for data analysis problems that includes several enhanced evolutionary modelling aspects.

7. Examples and applications of GP in data based structure identification

7.1 Regression

For demonstrating the use of our evolutionary machine learning approach for attacking regression problems we have generated a synthetic data set including 5 variables and 400 samples. This data was analyzed using the HeuristicModeler and a model was trained; this model is graphically shown in Fig. 7. There are several possibilities how to evaluate a regression model using the HeuristicModelAnalyzer: Apart from drawing the (original and estimated) values and a graphical representation of the formula as a structure tree, the average squared error can be calculated as well as an overview of the errors distribution (as exemplarily shown later in Fig. 11.



Anchor Formula: +[-[+([1*Var1(t)]*(2|[1,1*Var2(t)]))|[1*Var0(t)]]|[0,8*Var3(t)]]

Actual Formula: +[-[+([1*Var1(t)]*(2|[1,1*Var2(t)]))|[1*Var0(t)]]|[0,8*Var3(t)]]

Problem File: RegressionExample
C:\Documents and Settings\Swinkler\My Documents\HeuristicLab Documents\INS - RegressionExample.prb

Fitness value (evaluated on problem's training samples): 0,701106

Fitness value (evaluated on all samples): 0,674769

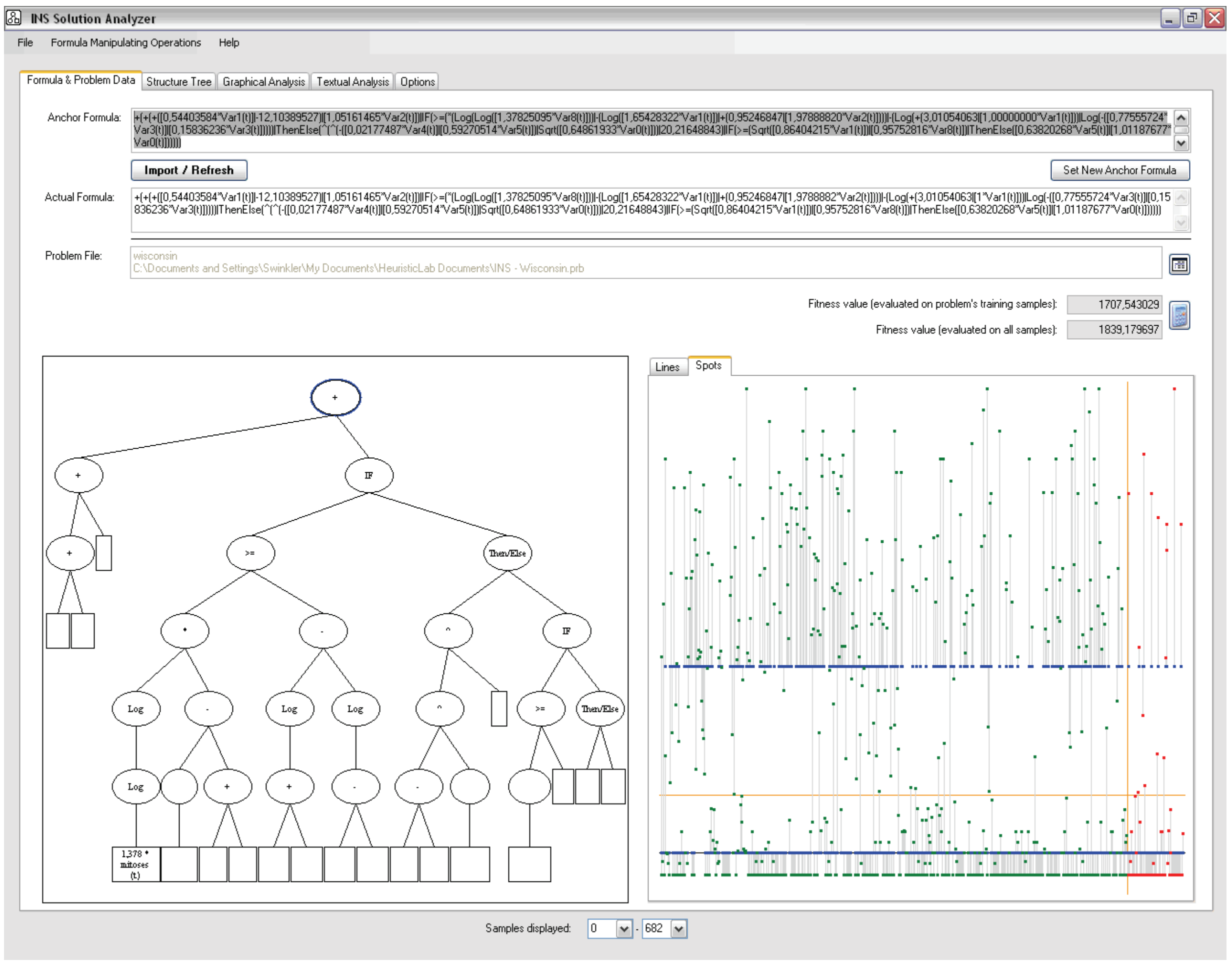
Samples displayed: 0 - 399

Fig. 7. A solution to a regression problem, analyzed using the HeuristicModelAnalyzer.

7.2 Classification

Several widely used benchmark classification datasets storing medical data (mainly survey records and diagnosis information) have already been analyzed using HeuristicModeler and HeuristicModelAnalyzer. In (Winkler et al., 2006b), (Winkler et al., 2006a) and (Winkler et al., 2006e) we have documented the results achieved for several medical classification benchmark problems, for example for the Wisconsin and the Thyroid datasets, which are parts of the UCI Machine Learning Repository (http://www.ics.uci.edu/~mlearn/). Summarizing the results documented in the publications mentioned above, GP-based training of classifiers is able to outperform other training methods (kNN classification, linear modeling and ANNs) especially on test data. There are several possibilities how to evaluate a classification model using the HeuristicModelAnalyzer:
Apart from drawing the (original and estimated) values and a graphical representation of the formula as a structure tree and calculating the average squared error, confusion matrices and (enhanced) receiver operating characteristics (ROC) curves can be generated. Furthermore, optimal thresholds are also identified automatically on the basis of a misclassification matrix storing information about how to weight misclassification dependent on the respective classes involved. This matrix is initially set so that all misclassifications are weighted equally; in various different applications it can be necessary to manipulate this weighting as it is, for example in the context of medical data analysis,

more critical misclassifying a diseased patient as not diseased than vice versa. In Fig. 8 we show a graphical representation of a solution for the Wisconsin classification problem that was generated using the HeuristicModeler and analyzed using the HeuristicModelAnalyzer. As confusion matrices are also frequently used for evaluating classifiers, these are also automatically displayed when analyzing a model using the HeuristicModelAnalyzer.



Anchor Formula: +(+(+([0,54403584*Var1(t)]-12,10389527)|[1,05161465*Var2(t)]|IF(>=(*(Log(Log([1,37825095*Var8(t)]))-(Log([1,65428322*Var1(t)]+(0,95246847|[1,97888820*Var2(t)])))-(Log(+(3,01054063|[1,00000000*Var1(t)]))|Log(-([0,77555724*Var3(t)]|[0,15836236*Var3(t)])))))ThenElse(^(^(-([0,02177487*Var4(t)]|[0,59270514*Var5(t)])|Sqrt([0,64861933*Var0(t)])|20,21648843)|IF(>=(Sqrt([0,86404215*Var1(t)])|[0,95752816*Var8(t)])|ThenElse([0,63820268*Var5(t)]|[1,01187677*Var0(t)])))))

Actual Formula: +(+(+([0,54403584*Var1(t)]-12,10389527)|[1,05161465*Var2(t)]|IF(>=(*(Log(Log([1,37825095*Var8(t)]))-(Log([1,65428322*Var1(t)]+(0,95246847|[1,9788882*Var2(t)])))-(Log(+(3,01054063|[1*Var1(t)]))|Log(-([0,77555724*Var3(t)]|[0,15836236*Var3(t)])))))ThenElse(^(^(-([0,02177487*Var4(t)]|[0,59270514*Var5(t)])|Sqrt([0,64861933*Var0(t)])|20,21648843)|IF(>=(Sqrt([0,86404215*Var1(t)])|[0,95752816*Var8(t)])|ThenElse([0,63820268*Var5(t)]|[1,01187677*Var0(t)])))))

Problem File: wisconsin
C:\Documents and Settings\Swinkler\My Documents\HeuristicLab Documents\INS - Wisconsin.prb

Fitness value (evaluated on problem's training samples): 1707,543029

Fitness value (evaluated on all samples): 1839,179697

Samples displayed: 0 - 682

Fig. 8. A solution for the Wisconsin classification problem, generated by the HeuristicModeler and analyzed using the HeuristicModelAnalyzer.

Of course, classification problems occur not only in medical data analysis, but for example also in the context of data based quality pre-assessment in steel production. In (Winkler et al., 2006f) we report on an analysis done within an enhanced data processing process in cooperation with a large-scale industrial partner in steel industry. It was shown successfully that GP based structure identification is able to identify relationships between process parameters and the quality of steel products; on the basis of these results, high quality classification pre-estimators for the quality of the final results were formed.

Last, but not least the HeuristicModelAnalyzer enables the evaluation of classifiers for multi-class classification problems on the basis of a multi-class extension of ROC curves. Basic ROC analysis provides a convenient graphical display of the trade-off between true and false positive classification rates for two class problems (Zweig & Vampell, 1993). In the context of two class classification, ROC curves are calculated as follows: For each possible threshold value discriminating two given classes, the numbers of true and false classifications for one of the classes are calculated. For example, if the two classes "true" and "false" are to be discriminated using a given classifier, a fixed set of equidistant thresholds is tested and the true positives (TP) and the false positives (FP) are counted for each of them. Each pair of TP and FP values produces a point of the ROC curve. The main idea of Multi-ROC charts as presented in (Winkler et al., 2006d) is that for each given class c_i the numbers of true and false classifications are calculated for each possible pair of thresholds between the classes c_{i-1} and c_i as well as between c_i and c_{i+1} (assuming that the n classes can be represented as real numbers and that $c_i < c_{i+1}$ holds for every $i \in [1,(n-1)]$). The resulting tuples of (FP,TP) values are stored in a matrix which can be plotted easily. This obviously yields a set of points which can be interpreted analog to the interpretation of "normal" ROC curves: the closer the point are located to the left upper corner, the higher is the quality of the classifier at hand. For getting sets of ROC curves instead of ROC points, an arbitrary threshold ta between the classes c_{i-1} and c_i is fixed and the FP and TP values for all possible thresholds tb between c_i and c_{i+1} are calculated. This produces one single ROC curve; it is executed for all possible values of ta. An example showing 10 ROC curves is given in Fig. 9; this MROC chart was generated for a classifier learned for a synthetical data set storing 2000 samples divided into 6 classes and is taken from (Winkler et al., 2006d).

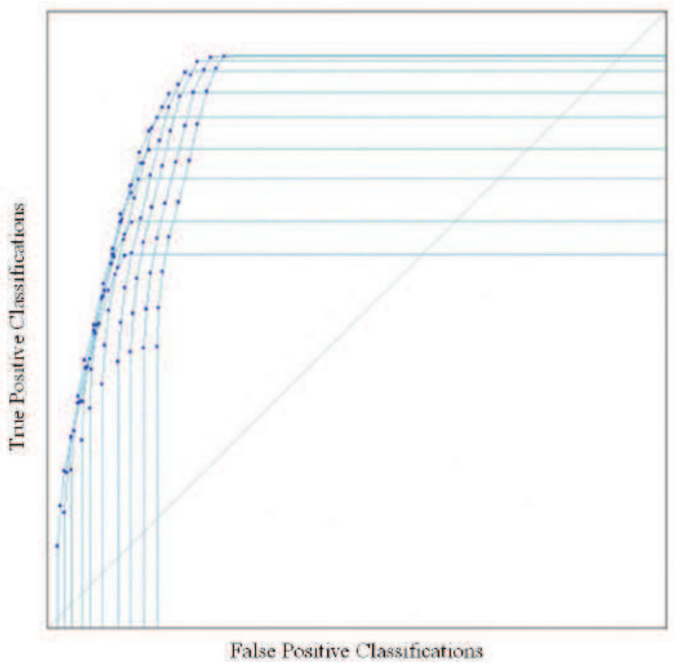
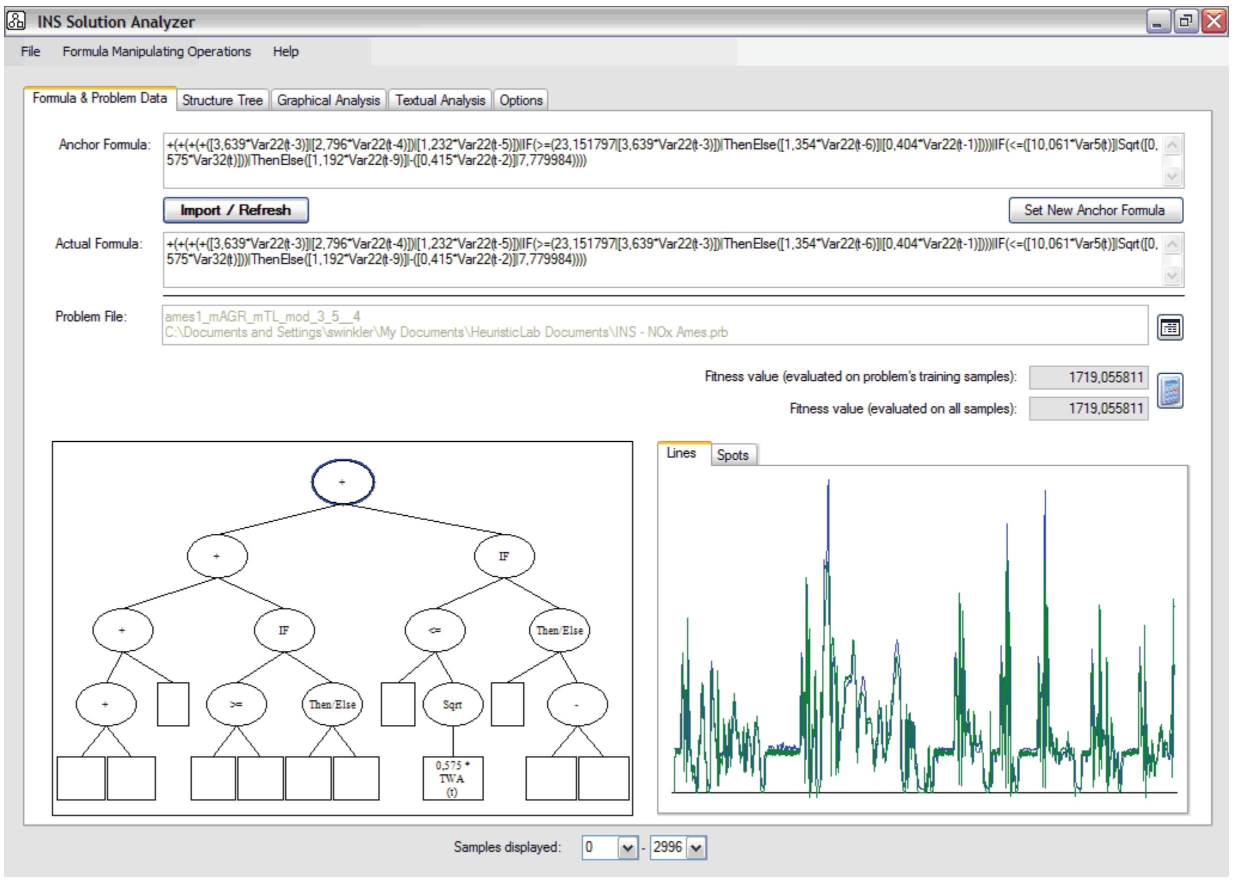


Fig. 9. An exemplary Multi-ROC chart.

7.3 Timeseries analysis

There is a lot of experience using the HeuristicModeler for solving time series problems on data recorded in the context of mechatronical systems. For example, in (Del Re et al., 2005) and (Winkler et al., 2005b) we report on models trained for the NO_x emissions of Diesel engines using the GP-based identification method incorporated in the HeuristicModeler. Fig. 10 and 11 show the evaluation of one of these models using the HeuristicModelAnalyzer: Apart from drawing the (original and estimated) values and a graphical representation of the formula as a structure tree, an overview of the errors distribution is given.



Anchor Formula: +(+(+(+([3,639*Var22(t-3)]|[2,796*Var22(t-4)])|[1,232*Var22(t-5)])|IF(>=(23,151797|[3,639*Var22(t-3)])|ThenElse([1,354*Var22(t-6)]|[0,404*Var22(t-1)])))|IF(<=([10,061*Var5(t)]|Sqrt([0,575*Var32(t)]))|ThenElse([1,192*Var22(t-9)]|-([0,415*Var22(t-2)]|7,779984))))

Actual Formula: +(+(+(+([3,639*Var22(t-3)]|[2,796*Var22(t-4)])|[1,232*Var22(t-5)])|IF(>=(23,151797|[3,639*Var22(t-3)])|ThenElse([1,354*Var22(t-6)]|[0,404*Var22(t-1)])))|IF(<=([10,061*Var5(t)]|Sqrt([0,575*Var32(t)]))|ThenElse([1,192*Var22(t-9)]|-([0,415*Var22(t-2)]|7,779984))))

Problem File: ames1_mAGR_mTL_mod_3_5__4
C:\Documents and Settings\swinkler\My Documents\HeuristicLab Documents\INS - NOx Ames.prb

Fitness value (evaluated on problem's training samples): 1719,055811

Fitness value (evaluated on all samples): 1719,055811

Samples displayed: 0 - 2996

Fig. 10. A model for the NO$_x$ emissions of a BMW Diesel engine, generated using the HeuristicModeler.

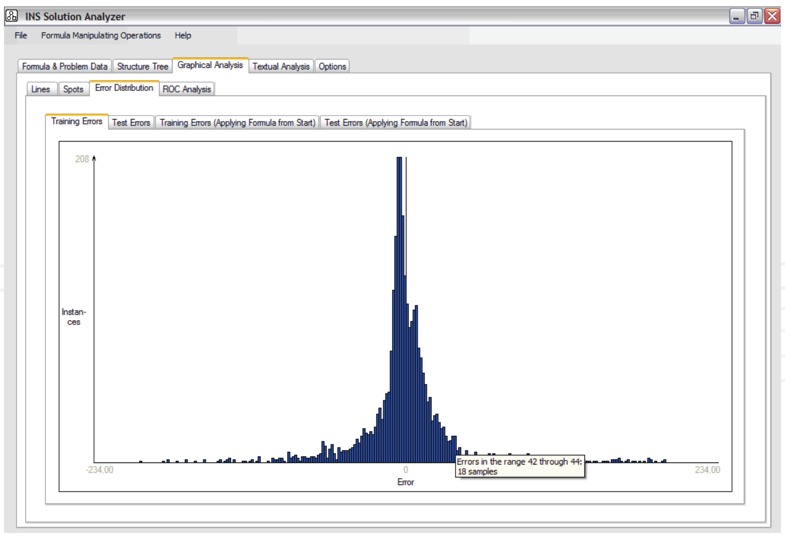


Fig. 11. Evaluation of the model shown in Figure 10.

8. Conclusion

In this paper we have described a multi-purpose machine learning approach based on various evolutionary computation concepts that is applicable for several data mining aspects in data driven systems identification. We have exemplarily shown how regression, classification and time series problems can be attacked using this algorithm. Especially in the context of analyzing time series problems of mechatronical systems as well as medical data sets we have already achieved very good results. Furthermore, we have also demonstrated how to analyze the results for data mining problems as well as selected aspects of the underlying enhanced evolutionary algorithm.

9. Acknowledgements

The work described in this paper was partially done within the Translational Research Program project L284-N04 "GP-Based Techniques for the Design of Virtual Sensors" sponsored by the Austrian Science Fund (FWF). This project is executed as a joint venture by the Upper Austrian University of Applied Sciences Hagenberg, Austria and the Johannes Kepler University Linz, Austria.

10. References

Affenzeller, M. (2001). Segregative Genetic Algorithms (SEGA): A Hybrid Superstructure Upwards Compatible to Genetic Algorithms for Retarding Premature Convergence. *International Journal of Computers, Systems and Signals (IJCSS),* Vol. 2, No. 1, (Feb. 2002) 18 -- 32, ISSN 1608-5655

Affenzeller, M. (2005). *Population Genetics and Evolutionary Computation: Theoretical and Practical Aspects,* Trauner Verlag, ISBN 3-85487-823-0, Linz, Austria

Affenzeller M & Wagner S. (2004). 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,* Vol. 10, 239--263, ISSN

Affenzeller M. & Wagner S. (2003). A Self-Adaptive Model for Selective Pressure Handling within the Theory of Genetic Algorithms, *Proceedings of EUROCAST 2003*, pp. 384-393, LNCS 2809, Las Palmas, Feb. 2003, Springer, Heidelberg

Beligiannis, G.N.; Skarlas, L.V.; Likothanassis, S.D.. & Perdikouri, K. (2003). Nonlinear Model Structure Identication of Complex Biomedical Data Using a Genetic Programming Based Technique, *Proceedings of IEEE International Symposium on Intelligent Signal Processing*.

Beyer, H. G. (1998) The Theory of Evolution Strategies. Springer-Verlag Berlin Heidelberg New York, 1998.

Box; G.E. & Jenkins, G.M. (1976). Time Series Analysis: Forecasting and Control. Holden-Day Inc., San Francisco, 1976.

Del Re, L.; Langthaler, P.; Furtmller, C.; Affenzeller, M. & Winkler, S. (2005). NO_x Virtual Sensor Based on Structure Identification and Global Optimization. Proceedings of the SAE World Congress 2005.

Dumitrescu, D.; Lazzerini, B.; Jain, L.C. & Dumitrescu A. (2000). *Evolutionary Computation*. CRC Press,2000.

Fogel, D.B. (1994). An introduction to simulated evolutionary optimization. *IEEE Trans. on Neural Network*, Vol.5, No. 1. 3--14, 1994.

Goldberg, D.E. (1989) *Genetic Alogorithms in Search, Optimization and Machine Learning*. Addison Wesley Longman, 1989.

Holland, J.H. (1975). *Adaption in Natural and Artificial Systems.* University of *Michigan* Press, 1975.

Kendall, S.M. & Ord, J.K. (1990). *Time Series*. Edward Arnold, London, 1990.

Koza, J.R. (1992). *Genetic Prohramming: On the Programming of Computers by means of Natural Selection*. The MIT Press, 1992.

Langdon, B. & Poli, R. (2002). *Foundations of Genetic Programming*. Springer-Verlag Berlin Heidelberg New York, 2002.

Michaliwicz, Z. (1996). *Genetic Algorithms + Data Structurs = Evolution Programs.* Springer-Verlag Berlin Heidelberg New York, 3. edition, 1996.

Mitchell, T.M. (2000). *Machine Learning*. McGraw-Hill, New York, 2000.

Rechenberg, I. (1973). *Evolutionsstrategie*. Friedrich Frommann Verlag, 1973.

Rodrguez-Vzquez, K. & Fleming, P.J. (2000). Use of Genetic Programming in the Identification of Rational Model Structures. *Third European Conference on Genetic Programming (EuroGP'2000)*, pp 181--192, 2000.

Schwefel, H.P. (1994). *Numerische Optimierung von Computer-Modellen mittels Evolutionsstrategie*. Birkhaeuser Verlag. Basel, 1994.

Tomassini M. (1995). A survey of genetic algorithms. Annual Reviews of Computational Physics, Vol.3: 87--118, 1995.

Wagner, S. & Affenzeller, M. (2005a). Heuristiclab: A generic and extensible optimization environment. Adaptiveand Natural Computing Algorithms – Proceedings of ICANNGA 2005, pp. 538 -- 541, 2005.

Wagner, S. & Affenzeller, M (2005b). SexualGA: Gender-specific selection for genetic algorithms. Proceedings of the 9th World Multi-Conference on Systemics, Cybernetics and Informatics (WMSCI), pages 76--81, 2005.

Winkler, S.; Affenzeller, M. & Wagner, S. (2005a). New methods for the identification of nonlinear model structures based upon genetic programming techniques. *Journal of Systems Science*, Vol.31, 5--13, 2005.

Winkler, S.; Affenzeller, M. & Wagner, S. (2006a). Advances in applying genetic programming to machine learning focussing on classi¯cation problems. Proceedings of the 20th IEEE International Parallel & Distributed Processing Symposium (IPDPS 2006), 2006.

Winkler, S.; Affenzeller, M. & Wagner, S. (2006b). Automatic data based patient classification using genetic programming. Cybernetics and Systems 2006, 1:251--256, 2006.

Winkler, S.; Affenzeller, M. & Wagner, S. (2006c). Heuristic Modeler: A Multi-Purpose Evolutionary Machine Learning Algorithm and its Applications in Medical Data Analysis. Proceedings of the International Mediterranean Modelling Multiconference I3M 2006, pp. 629--634, 2006.

Winkler, S.; Affenzeller, M. & Wagner, S. (2006d). Sets of Receiver Operating Characteristic Curves and their use in the Evaluation of Multi-class Classification. Proceedings of the Genetic and Evolutionary Computation Conference 2006 (GECCO2006), 2: pp. 1601--1602, 2006.

Winkler, S.; Affenzeller, M. & Wagner, S. (2006e). Using Enhanced Genetic Programming Techniques for Evolving Classiffers in the Context of Medical Diagnosis - an Empirical Study. Proceedings of the GECCO 2006 Workshop on Medical Applications of Genetic and Evolutionary Computation (MedGEC 2006), 2006.

Winkler, S.; Efendic,H.; Affenzeller, M.; Del Re, L. & Wagner, S. (2005b) On-Line Modeling Based on Genetic Programming. Proceedings of the 1st International Workshop on Automatic Learning and Real-Time (ALaRT'05), pp. 119--130, 2005.

Winkler, S.; Efendic,H.& Del Re, L. (2006f). Quality Pre-assesment in Steel Industry using Data Based Estimators. Proceedings of the IFAC Workshop MMM2006 on Automation in Mining, Mineral and Metal Industry, pp. 185--190, 2006.

Zweig, M. H. & Campbell, G. (1993). Receiver-Operating Characteristics (ROC) Plots: A Fundamental Evaluation. *Clinical Chemistry*, Vol. 39: 561--577, 1993.