Selection Pressure Driven Sliding Window Behavior in Genetic Programming Based Structure Identification *

Stephan Winkler[1], Michael Affenzeller[2], Stefan Wagner[2]

Upper Austrian University of Applied Sciences
[1] Research Center Hagenberg
[2] Department of Software Engineering
Softwarepark 11, 4232 Hagenberg, Austria
{stephan,michael,stefan}@heuristiclab.com

Abstract. Virtual sensors are a key element in many modern control and diagnosis systems, and their importance is continuously increasing; if there are no appropriate models available, virtual sensor design has to be based on data. Structure identification using Genetic Programming is a method whose ability to produce models of high quality has been shown in many theoretical contributions as well as empirical test reports. One of its most prominent shortcomings is relatively high runtime consumption; additionally, one often has to deal with problems such as overfitting and the selection of optimal models out of a pool of potential models that are able to reproduce the given training data.
In this article we present a sliding window approach that is applicable for Genetic Programming based structure identification; the selection pressure, a value measuring how hard it is to produce better models on the basis of the current population, is used for triggering the sliding window behavior. Furthermore, we demonstrate how this mechanism is able to reduce runtime consumption as well as to help finding even better models with respect to test data not considered by the training algorithm.

1 Genetic Programming Based Structure Identification

Virtual sensors are widely understood as calculation models that can be used instead of physical sensors; they are a key element in many modern control and diagnosis systems, and their importance is continuously increasing. In case no appropriate mathematical models are available (to the required precision), virtual sensor design must be based on data. In most such cases of systems identification, universal approximators (as for example Artificial Neural Networks) are commonly used, even though their limits are well known. Unlike these methods, Genetic Programming (GP) based techniques have been used successfully

* 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).

for solving data based structure identification problems in various different areas of data mining. Within the last years we have set up a further developed, fully automated and problem domain independent GP based structure identification framework that has been successfully used in the context of various different kinds of identification problems for example in mechatronics ([2], [10]), medical data analysis [8] and the analysis of steel production processes [9].

Basically, GP is based on the theory of Genetic Algorithms (GAs). A GA works with a set (population) of solution candidates. New individuals are created on the one hand by combining the genetic make-up of two previously selected solution candidates (by "crossover"), and on the other hand by mutating some individuals, which means that randomly chosen parts of genetic information are changed (which is normally applied on a minor ratio of the algorithm's population). Each individual is evaluated using a pre-defined fitness function.

Similar to GAs, GP also utilizes a population of solution candidates which evolves through many generations towards a solution using certain evolutionary operators and a selection scheme increasing better solutions' probability of passing on genetic information. 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 solving the optimization problem at hand. In the case of structure identification, solution candidates represent mathematical models (for example stored as formula structure trees). These are evaluated by applying the formulae to the given training data and comparing the so generated output to the original target data. 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. The left part of Fig. 1 visualizes how the GP cycle works: As in every evolutionary process, new individuals (in GP's case, new programs) are created and tested, and the fitter ones in the population succeed in creating children of their own; unfit ones die and are removed from the population [5].

As already stated before, the results achieved for various data based identification problems using GP are satisfying, but still there are several problems that are to be considered, the most obvious ones being on the one hand the rather high runtime consumption and on the other hand over-fitting (which means that the algorithm produces models that show a good fit on training data but perform badly on test data). Even though the GP-based approach has been shown to be much less likely to over-fit [8], still this is an issue that is always present in the context of data based modeling.

2 On-Line and Sliding Window Genetic Programming

The idea of sliding window behavior in computer science is not novel; in machine learning, drifting concepts are often handled by moving the scope (of either fixed or adaptive size) over the training data (see for example [7] or [3]). The main idea is the following: Instead of considering all training data for training models (in the case of GP, for evaluating the models produced), the algorithm initially

only considers a part of the data. Then, after executing learning routines on the basis of this part of the data, the range of samples under consideration is shifted by a certain offset. Thus, the window of samples considered is moved, it slides along the training data; this is why we are talking about sliding window behavior.

When it comes to GP based structure identification, sliding window approaches are not all too common; in general, the method is seen as a global optimization method working on a set of training samples (which are completely considered by the algorithm within the evaluation of solution candidates). On the contrary, GP is often even considered as an explicitely off-line, global optimization technique. Nevertheless, during research activities in the field of on-line systems identification [10], we discovered several surprising aspects. In general, on-line GP was able to identify models describing a Diesel engine's NO_x emissions remarkably fast; the even more astonishing fact was that these models were even less prone to over-fitting than those created using standard methods. After further test series and reconsidering the basic algorithmic processes, these facts did not seem to be surprising to us anymore: On the one hand, especially the fact that the environment, i.e. the training data currently considered by the algorithm, is not constant but rather changing during the execution of the training process, contributes positively to the models' quality, it obviously decreases the threat of overfitting. On the other hand, the interplay of a changing data basis and models created using different data also seems to be contributing in a positive way. As the on-line algorithm is executed and evaluates models using (new) current training data forgetting samples that were recorded in the beginning, those "old" data are really forgotten from the algorithm's point of view. Still, the models created on the basis of these old data are still present. The behavior that results out of this procedure is more or less that several possible models that explain the first part of the data are created, and as the scope is moved during the algorithm's execution, only those models are successful that are also able to explain "new" training data.

So, the most self-evident conclusion was that these benefits of online training should be transferred to off-line training using GP. Obviously, this directly leads us to sliding window techniques which are described in the following sections.

3 Selection Pressure as Window Moving Trigger

One of the most important problem independent concepts used in our implementation of GP-based structure identification is Offspring Selection [1], an enhanced selection model that has enabled Genetic Algorithms and Genetic Programming implementations to produce superior results for various kinds of optimization problems[1]. As in the case of conventional GAs or GP, offspring are generated by parent selection, crossover, and mutation. In a second (offspring) selection

[1] In fact, this procedure of eliminating offspring that do not meet the given requirements (i.e. that are assigned a quality value worse than their parents) is also integrated in the left part of Figure 1.

step (as it is used in our GP implementation), only those children become members of the next generation's population that outperform their own parents, all other ones are discarded. The algorithm therefore repeats the process of creating new children until the number of successful offspring is sufficient to create the next generation's population[2]. Within this selection model, selection pressure is defined as the ratio of generated candidates to the population size:

$$SelectionPressure = \frac{\mid Solution\ \ candidates\ \ created \mid}{\mid Successful\ \ solution\ \ candidates \mid}$$

The higher this values becomes, the more models have to be created and evaluated in order to produce enough models that are supposed to form the next generation's population. In other words, this selection pressure is a value giving a measure of how hard it is for the algorithm to produce a sufficient number of successful solution candidates. This simplified Offspring Selection model is schematically displayed in the right part of Figure 1.

The proposed idea is to initially reduce the amount of data that is available for the algorithm as identification data. As the identification process is executed, better and better models are created which leads to a rise of the selection pressure; as soon as the selection pressure reaches a predefined maximum value, the limits of the identification data are shifted and the algorithm goes on considering another part of the available identification data set. This procedure is then repeated until the actual training data scope has reached the end of the training data set available for the identification algorithm, i.e. when all data have been considered. By doing so, the algorithm is (following the considerations formulated in the previous section) even less exposed to over-fitting, and due to the fact that the models created are evaluated on much smaller data sets we also expect a significant decrease of runtime consumption.

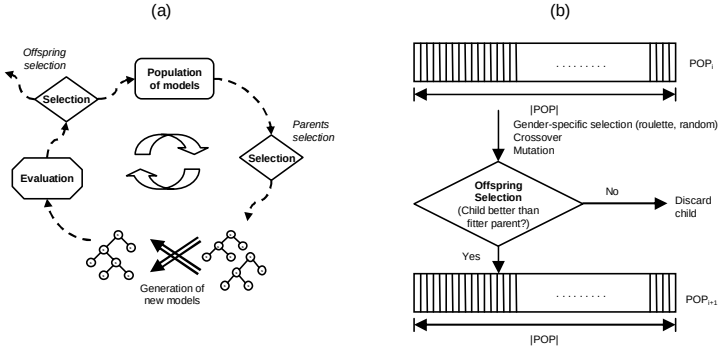

Fig. 1. (a) The Extended Genetic Programming Cycle including Offspring Selection; (b) Embedding a simplified version of Offspring Selection into the GP process.

[2] In fact, there are several aspects of Offspring Selection that are not explained here; a more detailed description can be found in [1].

Algorithm 1 The sliding window based GP structure identification process.

function Model = SlidingWindowGPStructId (TrainingData, FunctionalBasis, WindowStartSize, WindowStepSize, MaximumWindowSize, MaximumSelectionPressure1, MaximumSelectionPressure2)
  $index1 = 1$, $index2 = WindowStartSize - 1$
  $InitializeModelsPool$
  while $index2 <= Data.Length$ **do**
    while $CurrentSelectionPressure <= MaximumSelectionPressure1$ **do**
      Perform GP-based structure identification using the given $TrainingData$ in the interval $[index1; index2]$
    end while
    $index2 = Min((index2 + WindowStepWidth), Data.Length)$
    $index1 = Max((index2 - MaximumWindowSize + 1), 0)$
  end while
  $index2 = Data.Length$
  $index1 = Max((Data.Length - MaximumWindowSize + 1), 0)$
  while $CurrentSelectionPressure <= MaximumSelectionPressure2$ **do**
    Perform GP-based structure identification using the given $TrainingData$ in the interval $[index1; index2]$
  end while
return Current best model

In Algorithm 1 we give a sketch of the sliding window GP based structure identification process incorporating Offspring Selection. The standard GP parameters (as, for example, population size, mutation rate and crossover operator combinations) are hereby omitted; we only describe the sliding window specific process modifications. The $WindowStartSize$ parameter gives the initial size of the current data window; as soon as the current selection pressure reaches $MaximumSelectionPressure1$, the window is moved by $WindowStepSize$ samples; the $MaximumWindowSize$ parameter specifies the maximum size of the current training data scope. This procedure is repeated until the end of the data set is reached; in the end, the process stops as soon as the second maximum selection pressure parameter value is reached (which does not necessarily have to be the same as the first maximum selection pressure value).

4 Experiments

4.1 Experimental Setup, Data Basis

For testing the sliding window approach described here we have used the *HeuristicModeler*, a GP-based data mining approach implemented within the HeuristicLab [6], a paradigm-independent and extensible environment for heuristic optimization. In the following we will report on tests executed using the *Thyroid* data set, a widely used machine learning benchmark data set containing the results of medical measurements which were recorded while investigating patients poten-

tially suffering from hypotiroidism[3]. 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. In the following we are going to report on test results achieved using the first 80% of the data (containing 7,200 samples in total) as training data and the remaining 20% for testing the models created. We here state the quality of the classifiers created by the identification process using the mean squared error function for evaluating them.

4.2 Parameter Settings and Test Results

For testing the sliding window approach presented in this paper and also for comparing its ability to produce models of high quality we have tested the following 5 different GP-based data mining strategies characterized by their population size $|pop|$, maximum selection pressure values MSP (maximum selection pressure, $MSP1$ and $MSP2$ (maximum selction pressure values 1 and 2 as explained in the previous section) and relative values for sliding window parameters:

1. Standard-GP: $|pop| = 2000$, 1500 generations, no Offspring Selection.
2. GP including Offspring Selection: $|pop| = 1000$, $MSP = 200$
3. Sliding window GP: $|pop| = 1000$, $MSP1 = 50$, $MSP2 = 200$, sliding window: initial size 0.2, step width 0.1, maximum size 0.4
4. Sliding window GP: $|pop| = 1000$, $MSP1 = 50$, $MSP2 = 200$, sliding window: initial size 0.4, step width 0.2, maximum size 0.5
5. Sliding window GP: $|pop| = 1000$, $MSP1 = 20$, $MSP2 = 200$, sliding window: initial size 0.2, step width 0.05, maximum size 0.4

All tests were executed applying 15% mutation rate and a combination of random and roulette parent selection schemata. For each test scenario we have executed 5 independent test runs. In the following table we give average numbers of iterations and solutions evaluated as well as the quality of the models identified (average values as well as the quality of each test series' model showing the best fit on the complete training data set) with respect to (complete) training and test data. Please note that all variables were normalized independently (i.e. scaled linearily so that the resulting variables' mean values are equal to 0 and their standard deviations are exactly 1.0). The maximum size of models created by the training algorithm was set to 60, the maximum formula tree height to 8.

In Figure 2 we give two characteristic screenshots: In the left part the selection pressure progress of one of the test runs of test series (5) is displayed (with vertical gray lines indicating training data scope drifts: every time the selection pressure became greater than 20, the window was shifted); in the right part we show a graphical representation of the evaluation of the best classifier trained in test series (4). This result is indeed remarkable as it correctly classifies 98.46% of the training and 98.08% of the test samples. This model even outperforms those reported on in [8]; a detailed confusion matrix is given in Table 2.

[3] Further information about the data set used can be found on the UCI homepage `http://www.ics.uci.edu/~mlearn/`.

Table 1. Results of the tests executed for the *Thyroid* data set.

Test Scenario	Iterations (Average)	Solutions Evaluated (Average)	Speed Up	Model Quality on Training Data		on Test Data	
				Best Model	Average	Best Model	Average
1	1,500.00	3,000,000.00	1.00	0.316	0.410	0.381	0.444
2	64.40	2,717,678.80	1.10	0.155	0.283	0.251	0.341
3	65.60	3,199,551.00	2.40	0.166	0.193	0.219	0.233
4	59.80	1,925,755.40	3.18	0.166	0.246	0.199	0.310
5	62.60	2,483,116.60	3.10	0.125	0.173	0.220	0.252

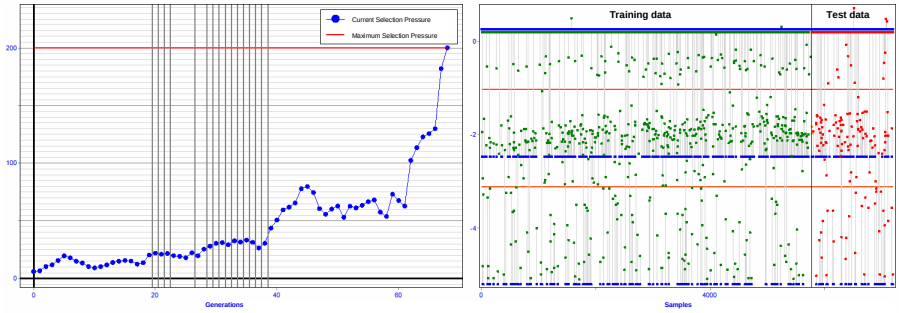

Fig. 2. Left: Selection pressure progress of the best test run of test series (5); Right: Graphical representation of the best test run of test series (4).

Obviously, as is summarized in Table 1, all GP methods using Offspring Selection perform significantly better that the standard implementation. Furthermore, the use of sliding window mechanisms here resulted in models that perform better on test data as well as in significant runtime reduction. Due to the fact that on the one hand not all training data but only the respective current data scopes are used in the sliding window test series and on the other hand the share of model evaluation in runtime consumption of GP based data mining is almost 100%, the algorithms are executed significantly faster: The respective speed up values range from 2.4 to almost 3.2.

Table 2. Analysis of the best model produced in test series (4) whose evaluation is displayed in Figure 2.

Original Class →		1	2	3	
Class Predicted	1	29 (2.06%)	0 (0.00%)	2 (0.14%)	
	2	5 (0.35%)	63 (4.47%)	19 (1.35%)	
	3	0 (0.00%)	1 (0.07%)	1290 (91.55%)	
Correctly Classified					1382 (98.08%)

5 Discussion, Summary

In this article we have presented a sliding window approach for data mining using evolutionary computation techniques, namely enhanced Genetic Programming. A further developed selection model (Offspring Selection), is used for determining the resulting selection pressure which is used for triggering the drift of the current training data scope. In the experimental part we have reported on a series of tests using a widely used classification benchmark problem for demonstrating the effects of the use of these enhanced aspects. It has been shown that it is possible to reduce the algorithm's runtime as well as to increase the models' test quality when applying the sliding window mechanism presented here.

Additional to further test series, more detailed analysis of this method is still needed. For example, the effects of this drifting mechanism on the genetic diversity are to be analyzed; we will report on this analysis in [11].

References

1. Affenzeller, M., Wagner, S.: Offspring Selection: A New Self-Adaptive Selection Scheme for Genetic Algorithms. Adaptive and Natural Computing Algorithms (2005) 218–221
2. Alberer, D., Del Re, L., Winkler, S., Langthaler, P.: Virtual Sensor Design of Particulate and Nitric Oxide Emissions in a DI Diesel Engine. Proceedings of the 7^{th} International Conference on Engines for Automobile ICE 2005 (2005) paper nr. 2005-24-063
3. Hulten, G., Spencer, L., Domingos, P.: Mining Time-Changing Data Streams. In: Proceedings of the 7^{th} ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2001) 97–106
4. Koza, J.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. The MIT Press (1992)
5. Langdon, W., Poli, R.: Foundations of Genetic Programming. Springer Verlag, Berlin Heidelberg New York (2002)
6. Wagner, S., Affenzeller, S.: Heuristiclab: A Generic and Extensible Optimization Environment. Adaptive and Natural Computing Algorithms (2005) 538–541
7. Widmer, G., Kubat, M.: Learning in the Presence of Concept Drift and Hidden Contexts. Machine Learning 23(2) (1996) 69–101
8. Winkler, S., Affenzeller, M., Wagner, S.: Using Enhanced Genetic Programming Techniques for Evolving Classifiers in the Context of Medical Diagnosis - An Empirical Study. GECCO 2006 Workshop on Medical Applications of Genetic and Evolutionary Computation (MedGEC 2006). ACM (2006) paper nr. WKSP115
9. Winkler, S., Efendic, H., Del Re, L.: Quality Pre-Assesment in Steel Industry Using Data Based Estimators. In: S. Cierpisz, K. Miskiewicz, A. Heyduk (eds.): Proceedings of the MMM 2006 Workshop on Automation in Mining, Mineral and Metal Industry. International Federation for Automatic Control (2006) 185–190
10. Winkler, S., Efendic, H., Affenzeller, M., Del Re, L., Wagner, S.: On-Line Modeling Based on Genetic Programming. Int. J. on Intelligent Systems Technologies and Applications, Vol. 2, NOs. 2/3. Inderscience Publishers (2007) 255–270
11. Winkler, S., Affenzeller, M., Wagner, S.: Genetic Diversity in Systems Identification Based on Extended Genetic Programming. Accepted to be published in Proceedings of the 16^{th} International Conference on Systems Science (2007)