Model Complexity Control in Straight Line Program Genetic Programming

César L. Alonso[1], José Luis Montaña[2] and Cruz Enrique Borges[3]

[1]*Centro de Inteligencia Artificial, Universidad de Oviedo ,Campus de Gijón, 33271 Gijón, Spain*
[2]*Departamento de Matemáticas, Estadística y Computación, Universidad de Cantabria, 39005 Santander, Spain*
[3]*DeustoTech (Energy Unit), Universidad de Deust, 48007 Bilbao, Spain*

Keywords: Genetic Programming, Straight Line Program, Pfaffian Operator, Symbolic Regression.

Abstract: In this paper we propose a tool for controlling the complexity of Genetic Programming models. The tool is supported by the theory of Vapnik-Chervonekis dimension (VCD) and is combined with a novel representation of models named straight line program. Experimental results, implemented on conventional algebraic structures (such as polynomials) and real problems, show that the empirical risk, penalized by suitable upper bounds for the Vapnik-Chervonenkis dimension, gives a generalization error smaller than the use of statistical conventional techniques such as Bayesian or Akaike information criteria.

1 INTRODUCTION

Inductive inference from examples is one of the most studied problems in Artificial Intelligence and has been addressed for many years using different techniques. Among them are included statistical methods such as inference techniques, regression and decision trees and other machine learning methods like neuronal networks and support vector machines((Tenebaum et al., 2006), (Angluin and Smith, 1983), (Gori et al., 1998), (Shaoning and Kasabov, 2004)).

In the last two decades, genetic programming (GP) has been applied to solving problems of inductive learning with some remarkable success ((Tackett and Carmi, 1994), (Oakley, 1994), (Poli and Cagnoni, 1997), (Nikolaev and Iba, 2001)). The general procedure involves the evolution of populations of data structures that represent models for the target function. In the evolutive process, the fitness function for evaluating the population measures some empirical error between the empirical value of the target function and the value of the considered individual over the sample set. Usually this fitness function must be regularized with some term that depends on the complexity of the model. Identifying optimal ways to measure the complexity of the model is one of the main goals in the process of regularization.

Most of the work devoted to develop GP strate-

gies for solving inductive learning problems makes use of the GP-trees as data structures for representing programs (Koza, 1992). We have proposed a new data structure named straight line program (slp) to deal with the problem of learning by examples in the framework of genetic programming. The slp has a good performance in solving symbolic regression problem instances as shown in (see (Alonso et al., 2008)). A slp consists of a finite sequence of computational assignments. Each assignment is obtained by applying some function (selected from a given set) to a set of arguments that can be variables, constants or pre-computed results. The slp structure can describe complex computable functions using a few amount of computational resources than GP-trees. The key point for explaining this feature is the ability of slps for reusing previously computed results during the evaluation process. Another advantage with respect to trees is that the slp structure can describe multivariate functions by selecting a number of assignments as the output set. Hence one single slp has the same representation capacity as a forest of trees. We study the practical performance of ad-hoc recombination operators for slps and we apply the slp- based GP approach to regression. In addition we study the Vapnik-Chervonekis dimension of slps representing models. We consider families of slp's constructed from a set of Pfaffian functions. Pfaffian functions are solutions of triangular systems of first order partial differen-

tial equations with polynomial coefficients. As examples, polynomials, exponential functions, trigonometric functions on some particular intervals and, in general, analytic algebraic functions are Pfaffian. The main outcome of this work is a penalty term for the fitness function of a genetic programming strategy based on slp's to solve inductive learning problems. Experimental results point out that the slp structure, if suitably regularized, may result in a robust tool for supervised learning.

2 STRAIGHT LINE PROGRAM GENETIC PROGRAMMING

Straight line programs have a large history in the field of Computational Algebra. A particular class of straight line programs, known in the literature as arithmetic circuits, constitutes the underlying computation model in Algebraic Complexity Theory ((Burguisser et al., 1997)). Arithmetic circuits with the standard arithmetic operations $\{+, -, *, /\}$ are the natural model of computation for studying the computational complexity of algorithms solving problems with an algebraic flavor. They have been used in linear algebra problems ((Berkowitz, 1984)), in quantifier elimination ((Heintz et al., 1990)) and in algebraic geometry ((Giusti and Heinz, 1993) and (Giusti et al., 1997)). Also, slp's constitute a promising alternative to the trees in the field of Genetic Programming (see (Alonso et al., 2008)). The formal definition of the straight line program structure is as follows: Let $F = \{f_1, \ldots, f_n\}$ be a set of functions, where f_i has arity a_i, for $1 \le i \le n$, and let $T = \{t_1, \ldots, t_m\}$ be a set of terminals. A *straight line program (slp)* over F and T is a finite sequence of computational instructions $\Gamma = \{I_1, \ldots, I_l\}$ where

$$I_k \equiv u_k := f_{j_k}(\alpha_1, \ldots, \alpha_{a_{j_k}}); \text{ with } f_{j_k} \in F,$$

$\alpha_i \in T$ for all i if $k = 1$ and $\alpha_i \in T \cup \{u_1, \ldots, u_{k-1}\}$ for $1 < k \le l$.

Terminal set T is of the form $T = V \cup C$, where $V = \{x_1, \ldots, x_n\}$ is a finite set of variables and $C = \{c_1, \ldots, c_q\}$ is a finite set of constants. The number of instructions l is the *length* of Γ.

Note that if we consider the *slp* Γ as the code of a program, then a new variable u_i is introduced at each instruction I_i. We will denote by $\Gamma = \{u_1, \ldots, u_l\}$ a slp. Each of the non-terminal variables u_i can be considered as an expression over the set of terminals T constructed by a sequence of recursive compositions from the set of functions F. The set of all slp's over F and T is denoted by $SLP(F, T)$.

An output set of a slp $\Gamma = \{u_1, \ldots, u_l\}$ is any set of non-terminal variables of Γ, that is, $O(\Gamma) = \{u_{i_1}, \ldots, u_{i_t}\}$, $i_1 < \cdots < i_t$. Provided that $V = \{x_1, \ldots, x_p\} \subset T$ is the set of terminal variables, the function computed by Γ, denoted by $\Phi_\Gamma : I^p \to O^t$, is defined recursively in the natural way and satisfies $\Phi_\Gamma(a_1, \ldots, a_p) = (b_1, \ldots, b_t)$, where b_j stands for the value of the expression over V of the non-terminal variable u_{i_j} when we replace the variable x_k with a_k; $1 \le k \le p$.

For computing the initial population, the well known methods for trees (see (Koza, 1992)) can be easily adapted to slp's. In order to compute the fitness function in a GP process to solve a particular problem, the computation of the function Φ_Γ, considering its own definition, would be often necessary.

2.1 SLP Crossover and Mutation

For slp-GP, 1-point crossover and in general k-point crossover are easily defined. However, a new specific crossover operation that produces another type of information exchange between the two selected parents, has been designed. The objective is to carry subexpressions from one parent to the other. A subexpression is captured by an instruction u_i and all the instructions that are used to compute the expression over the set of terminals represented by u_i. Now follows a description of this crossover with a clarifying example.

Given two slp's, Γ_1 and Γ_2, first a position k en Γ_1 is randomly selected. Let S_{u_k} be the set of all instructions related to the computation of the subexpression associated to node u_k. We obtain the first offspring by randomly selecting an allowed position t in Γ_2 and making the substitution of a part of its instructions by those instructions in S_{u_k} suitably renamed. For the second offspring we symmetrically repeat the strategy.

Example: Consider $F = \{*, +\}$, $L = 5$ and $T = \{x, y\}$. Let Γ_1 and Γ_2 be the following two slp's:

$$\Gamma_1 \equiv \begin{cases} u_1 := x + y \\ u_2 := u_1 * u_1 \\ u_3 := u_1 * x \\ u_4 := u_3 + u_2 \\ u_5 := u_3 * u_2 \end{cases} \qquad \Gamma_2 \equiv \begin{cases} u_1 := x * x \\ u_2 := u_1 + y \\ u_3 := u_1 + x \\ u_4 := u_2 * x \\ u_5 := u_1 + u_4 \end{cases}$$

If $k = 3$ then $S_{u_3} = \{u_1, u_3\}$, and t must be selected in $\{2, \ldots, 5\}$. If for instance $t = 3$, then the first offspring is as follows.

$$\Gamma_1' \equiv \begin{cases} u_1 := x * x \\ \mathbf{u_2 := x + y} \\ \mathbf{u_3 := u_2 * x} \\ u_4 := u_2 * x \\ u_5 := u_1 + u_4 \end{cases}$$

For the second offspring, if the selected position in Γ_2 is $k' = 4$, then $S_{u_4} = \{u_1, u_2, u_4\}$. Now if $t = 5$, then the offspring will be

$$\Gamma'_2 \equiv \begin{cases} u_1 := x + y \\ u_2 := u_1 * u_1 \\ \mathbf{u_3 := x * x} \\ \mathbf{u_4 := u_3 + y} \\ \mathbf{u_5 := u_4 * x} \end{cases}$$

The mutation operation in the slp structure consists of a change in one of the instructions. This change can be either the substitution of the complete instruction by another one randomly generated, or a little modification of just one of the arguments of the function in F that defines the instruction.

3 PFAFFIAN FUNCTIONS AND VCD OF FORMULAS

In this section we introduce some tools concerning the geometry of sets defined by boolean combinations of sign conditions over Pfaffian functions (semi-Pfaffian sets in the mathematical literature). A complete survey on the subject is due to Gabrielov and Vorobjov ((Gabrielov and Vorobjov, 2004)).

Definition: Let $U \subset \mathbf{R}^n$ be an open domain. A *Pfaffian chain* of length $q \geq 1$ and degree $D \geq 1$ in U is a sequence of real analytic functions $f_1, \ldots, f_q$ in U satisfying a system of differential equations

$$\frac{\partial f_i}{\partial x_j} = P_{i,j}(\mathbf{x}, f_1(\mathbf{x}), \ldots, f_i(\mathbf{x})) \qquad (1)$$

for $i = 1, \ldots q$ where $P_{i,j} \in \mathbf{R}[\mathbf{x}, y_1, \ldots, y_i]$ are polynomials of degree at most D and $\mathbf{x} = x_1, \ldots, x_n$.

A function f on U is called a Pfaffian of order q and degree (D, d) if

$$f(\mathbf{x}) = P(\mathbf{x}, f_1(\mathbf{x}), \ldots, f_q(\mathbf{x})) \qquad (2)$$

where $P \in \mathbf{R}[\mathbf{x}, y_1, \ldots, y_q]$ is a polynomial of degree at most $d \geq 1$ and $f_1, \ldots, f_q$ is a Pfaffian chain of length q and degree D.

The following functions are Pfaffian: $sin(x)$, defined on the interval $(-\pi + 2\pi r, \pi + 2\pi r)$; $tan(x)$, defined on the interval $(-\pi/2 + \pi r, \pi/2 + \pi r)$; e^x defined in $\mathbf{R}$; $log\, x$ defined on $x > 0$; $1/x$ defined on $x \neq 0$. $\sqrt{x}$ defined on $x \geq 0$. More generally, analytic algebraic functions are Pfaffian.

Definition: Let $\mathcal{F}$ be a class of subsets of a set X. We say that $\mathcal{F}$ shatters a set $A \subset X$ if for every subset $E \subset A$ there exists $S \in \mathcal{F}$ such that $E = S \cap A$. The VCD of $\mathcal{F}$ is the cardinality of the largest set that is shattered by $\mathcal{F}$.

Next we announce an upper bound for the VCD of a family of concept classes whose membership tests are computed by straight line programs involving Pfaffian operators over the real numbers. An important new issue is that we do not consider an upper bound for the length of the slp's. In previous results about VCD of programs or families of computation trees, a time bound approximated by the number of steps of the program execution or by the height of the computation tree is needed ((Golberg and Jerrum, 1995)). In our case we only need a bound for the number of the non-scalar slp's instructions. Those are instructions involving operations which are not in $\{+, -\}$.

A rough estimation of the *VC* dimension of slps using Pfaffian operators can be obtained computing the number of free parameters in families of slps with bounded non-scalar complexity. To do this let $T = \{t_1, \ldots, t_n\}$ be a set of terminals and let $F = \{+, -*, /, sign\} \cup \{f_1, \ldots, f_q\}$ be a set of functions, where the elements f_i constitute a Pfaffian chain of length q with arities bounded by A and the *sign* function is defined as $sign(x) = 1$ if $x > 0$ and 0 otherwise.

Let $\Gamma_{n,L}$ be the collection of $slp's$ Γ over F and T using at most L non-scalar operations and a free number of scalar operations. Then, the number of free parameters of a universal slp Γ_U that parameterizes the elements of the family $\Gamma_{n,L}$ is exactly:

$$N := L[3 + q + A(n + \frac{L-1}{2}) + 1] + n \qquad (3)$$

The proof is as follows. Introduce a set of parameters α, β and γ taking values in Z^k for a suitable natural number k, such that each slp in the family can be obtained specializing the parameters. For this purpose we define $u_{-n+m} = t_m$, for $1 \leq m \leq n$. Note that any non-scalar assignment u_i, $1 \leq i \leq L$ in a slp Γ belonging to $\Gamma_{n,L}$ is a function of $t = (t_1, \ldots, t_n)$ that can be parameterized as follows:
$u_i = U_i(\alpha, \beta, \gamma)(t) =$

$$\gamma^i_{-n}[\alpha^i_{-n}(\sum_{j=-n+1}^{i-1} \alpha_j{}^{i_1} u_j) * (\sum_{j=-n+1}^{i-1} \alpha_j{}^{i_2} u_j) +$$

$$+ (1 - \alpha^i_{-n})[\beta^i_{-n} \frac{\sum_{j=-n+1}^{i-1} \alpha_j{}^{i_1} u_j}{\sum_{j=-n+1}^{i-1} \alpha_j{}^{i_2} u_j} +$$

$$+ (1 - \beta^i_{-n}) sgn(\sum_{j=-n+1}^{i-1} \alpha_j{}^{i_1} u_j)]] +$$

$$+ (1 - \gamma^i_{-n})[\sum_{k=1}^{q} \gamma'_k f_k(\sum_{j=-n+1}^{i-1} \alpha_j{}^{i_1} u_j, \ldots, \sum_{j=-n+1}^{i-1} \alpha_j{}^{i_A} u_j)]$$

Now considering the last assignment as the output set of the slp Γ, this last assignment is parameterized as:

$$U = \sum_{j=-n+1}^{L} \alpha_j u_j$$

where u_j, $1 \le j \le L$ are the non-scalar assignments.

Finally counting the number of introduced parameters we will obtain Equation 3. The estimation given in Equation 3 can be converted, after certain algebraic manipulations, into an upper bound using theory of Pfaffian operators (see (Gabrielov and Vorobjov, 2004)). We omit the proof due to lack of space.

Main Theorem. Let $\Gamma_{L,n}$ the set of slps with n variables, at most L non-scalar operations, using operators in F that contains the operations $\{+, -, *, /, sign\}$ and Pfaffian operations f, where each f belongs to a fixed Pfaffian chain $\{f_1, \ldots, f_q\}$ of length q and degree $D \ge 2$. Let N be as in Equation 3. Then, the Vapnik-Chervonenkis dimension of $C_{k,n}$ is in the class:

$$O((q(N+n))^2 + (N+n)(L+q)$$
$$log_2((N+n)(L+1)(4+D))) \qquad (4)$$

Simplification. If we consider as constants parameters n, q, D and d, the VCD of the class is at most $O(L^4)$. This quantity gives an idea of the asymptotic maximum order of VCD of common classes of *GP*-models. We point out that this quantity is an upper bound and, possibly, far from being an optimal bound, but it can be used as starting point in further experimental developments.

4 MODEL SELECTION CRITERION

In supervised learning problems like regression and classification a considerable amount of effort has been done for obtaining good generalization error bounds. The results by Vapnik (see (Vapnik, 1998)) state the following error bound:

$$\varepsilon(f) \le \varepsilon_m(f) + \sqrt{\frac{h(log(2m/h)+1) - log(\eta/4)}{m}},$$
$$(5)$$

where h must be substituted by the upper bound of the VCD of the hypothesis class that contains the model f, η is the probability that the error bound is violated and m is the sample size. As usual in this context $\varepsilon(f)$ and $\varepsilon_m(f)$ stand, respectively, for the true mean square error and the empirical mean square error of the model f.

In our case, f will be represented by a straight line program $\Gamma \in SLP(F,T)$ where T contains n variables and F contains the operations on real numbers

$\{+, -, *, /, sign\}$ and Pfaffian operations over the reals. Note that the sets F and T are invariants throughout the model selection process. Hence, the search space of models forms a nested structure:

$$C_1 \subset C_2 \subset \cdots \subset C_L \subset \cdots$$

where C_L represents the class of slp's in $SLP(F,T)$ that have at most L non-scalar instructions. In this situation we will finally choose the model that minimizes the right side of Equation 5.

5 EXPERIMENTAL RESULTS

In this section we present the obtained results after an experimental phase in which symbolic regression problem instances were solved using the selection criterion described in the previous section. Our proposal is to consider straight line programs with Pfaffian instructions as the structure that represents the model. Then a GP algorithm is executed considering the recombination operators for slp's described in Section 2 and with fitness regularization function expressed in Equation 5. So we propose a model estimation via structural risk minimization (SRM). For the complexity measure h of the model, we will use the VCD bound in 3.

We will consider additive gaussian noise in the sample set $z = (x_i, y_i)_{1 \le i \le m}$. Hence, for a target function g, the sample set verifies: $y_i = g(x_i) + \varepsilon$, where ε is independent and identically distributed (i.i.d.) zero mean random error.

We will compare the effectiveness of the VCD fitness regularization method (VCD-SRM) with two well known representative statistical methods with different penalization terms:

- Akaike Information Criterion (AIC) which is as follows (see (Akaike, 1970)):

$$\varepsilon(f) = \varepsilon_m(f) + \frac{2h}{m}\sigma^2 \qquad (6)$$

- Bayesian Information Criterion (BIC) (see (Bernadro and Smith, 1994)):

$$\varepsilon(f) = \varepsilon_m(f) + (ln\, m)\frac{h}{m}\sigma^2 \qquad (7)$$

In the above expressions h stands for the number of free parameters of the model (Equation 3).

For measuring the quality of the final selected model, we have considered a new set of unseen points, generated without noise from the target function. This new set of examples is known as the test set or validation set. So, let $(x_i, y_i)_{1 \le i \le n_{test}}$ a validation set for the target function $g(x)$ (i.e. $y_i = g(x_i)$) and let $f(x)$

be the model estimated from the training data. Then the prediction risk $\varepsilon_{n_{test}}$ is defined by the mean square error between the values of f and the true values of the target function g over the validation set:

$$\varepsilon_{n_{test}} = \frac{1}{n_{test}} \sum_{i=1}^{n_{test}} (f(x_i) - y_i)^2 \qquad (8)$$

5.1 First Experiment

For the first experiment we have considered a set of 500 multivariate polynomials with real coefficients whose degrees are bounded by 5. The number of variables varies from 1 to 5 with 100 polynomials for each case. A second experiment was also performed solving symbolic regression problem instances associated to a new set of specific multivariate functions Finally a third experiment was performed considering some well known real benchmark problems. In all cases, when the GP process finishes, the best individual is selected as the proposed model for the corresponding target function.

We shall denote the set of polynomials as $P_n^R[X]$ with $X = (x_1, \ldots, x_n)$, $1 \le n \le 5$ and $x_i \in [-1,1]$ $\forall i$. The individuals are slp's over $F = \{+, -, *, /, sqrt, sin, cos, exp\}$. In order to avoid errors generated by divisions by zero, instead of the traditional division we will use in our computation the operation usually named "protected division", that returns 1 if the denominator is zero. Besides the variables x_i, the terminal set also includes five constants c_i, $1 \le i \le 5$, randomly generated in $[-1,1]$. Observe that although the target functions are polynomials, our set F not only contains the operators of sum, difference and product, but also contains other Pfaffian functions. This situation increments considerably the search space. Nevertheless, note that in a real problem situation usually we do not know if the target function is a polynomial or not.

The parameters for the GP process are the following: population size $M = 200$, probability of crossover $p_c = 0,9$, probability of mutation $p_m = 0,05$, and tournament selection of size 5. The real length of the slp's in the population is bounded by 40. Elitism and a particular generational replacement are used. In this sense, the offsprings do not necessarily replace their parents. After a crossover we have four individuals: two parents and two offsprings. We select the two best individuals with different fitness values. The motivation is to prevent premature convergence and to maintain diversity in the population.

As we are considering multivariate polynomials as target functions, the difficulty of the problem instance increases with the number of variables. Hence, to vary the size of the sample set as a function of
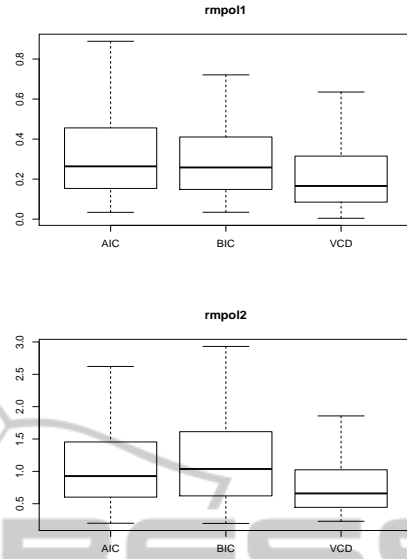

Figure 1: Empirical distribution of the executions, for the univariate and bivariate polynomials.

the number of variables is a reasonable decission. Note that an upper bound for the number of monomials in a polynomial with n variables and degree d is $4 \cdot d^{n+1}$ and this is also a quite good estimation for a lower bound of the size of the sample set. Thus, in our case we have considered sample sets of size $4 \cdot 5^{n+1}$, $1 \le n \le 5$. In this experiment one execution for each strategy has been performed over the 500 generated target functions. In every execution the process finishes after 250 generations were completed. Finally, the validation set consists of a number of unseen points that is equal to two times the size of the sample set.

Figures 1 and 2 represent the empirical distribution of the executions of the three compared strategies over the sets of polynomials. We have separated the polynomial sets by the number of variables, from one to five. These empirical distributions are displayed using standard box plot notation with marks at the best execution, 25%, 50%, 75% and the worst execution, always considering the prediction risk of the selected model, represented on the y-axis and defined by the mean square error between the values of the model and the true values of the target function over the validation set. We also include tables that show means and variances as well as the prediction risk of the best obtained model for each method.

As we can see from the above figures and tables, it seems that VCD regularization performs better than the well known regularization methods AIC and BIC. This is more clear for the polynomials up to three variables and not so clear for the rest of the polynomial
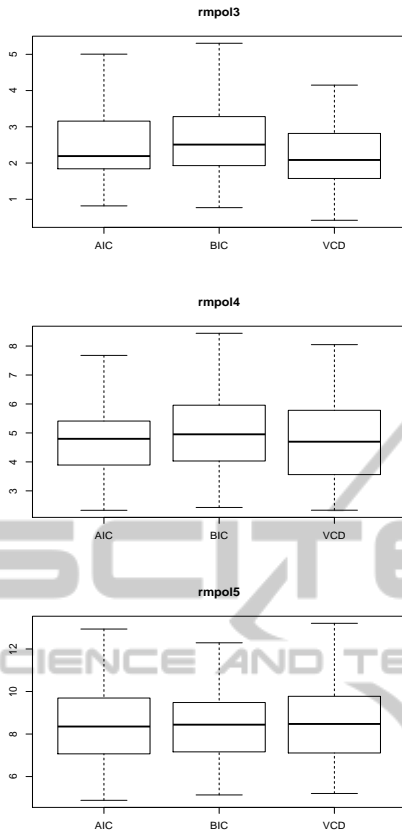
Figure 2: Empirical distributions of the executions, for the multivariate polynomials with 3, 4, and 5 variables.

Table 1: Values of means and variances.

$P_1^R[X]$	μ	σ	$P_2^R[X]$	μ	σ
AIC	0.39	0.42	AIC	1.17	0.86
BIC	0.38	0.41	BIC	1.25	0.85
VCD	0.24	0.27	VCD	0.82	0.54
$P_3^R[X]$	μ	σ	$P_4^R[X]$	μ	σ
AIC	2.62	1.28	AIC	4.79	1.49
BIC	2.89	1.57	BIC	5.08	1.74
VCD	2.24	0.85	VCD	4.76	1.43
$P_5^R[X]$	μ	σ			
AIC	8.51	2			
BIC	8.63	2.16			
VCD	8.63	2.17			

sets. This could be because for polynomials with four and five variables, as they constitute more complex problem instances, it would be necessary a large number of generations in the evolutive process. In order to confirm the comparative results of the studied strategies we have made crossed statistical hypothesis tests. The obtained results are showed in Table 2. Roughly speaking, the null-hypothesis in each test with asso-

Table 2: Results of the crossed statistical hypothesis tests about the comparative quality of the studied strategies.

$P_1^R[X]$	AIC	BIC	VCD
AIC		0.91	1
BIC	0.30		1
VCD	$3.93 \cdot 10^{-4}$	$1.1 \cdot 10^{-3}$	
$P_2^R[X]$	AIC	BIC	VCD
AIC		$3.91 \cdot 10^{-2}$	1
BIC	0.96		1
VCD	$1.86 \cdot 10^{-5}$	$1.12 \cdot 10^{-7}$	
$P_3^R[X]$	AIC	BIC	VCD
AIC		$7.7 \cdot 10^{-2}$	0.99
BIC	0.99		1
VCD	$1.2 \cdot 10^{-2}$	$3.93 \cdot 10^{-4}$	
$P_4^R[X]$	AIC	BIC	VCD
AIC		0.12	0.90
BIC	0.77		0.99
VCD	0.27	$2.7 \cdot 10^{-3}$	
$P_5^R[X]$	AIC	BIC	VCD
AIC		0.44	0.52
BIC	0.77		0.78
VCD	0.61	0.37	

Table 3: Prediction risk of the model obtained from the best execution.

Instance	AIC	BIC	VCD
$P_1^R[X]$	$3.40 \cdot 10^{-2}$	$3.44 \cdot 10^{-2}$	$4.32 \cdot 10^{-3}$
$P_2^R[X]$	0.20	0.19	0.22
$P_3^R[X]$	0.82	0.77	0.42
$P_4^R[X]$	2.33	2.43	2.33
$P_5^R[X]$	4.88	5.13	5.20

ciated pair (i, j) is that strategy i is not better than strategy j. Hence if value a_{ij} in Table 2 is less than a significance value α, we can reject the corresponding null-hypothesis.

Taking into account the results of the crossed statistical hypothesis tests with a significance value $\alpha = 0.05$, we can confirm that our proposed regularization method based on the VC dimension of families of SLP's is the best of the studied strategies for the considered sets of multivariate polynomials.

5.2 Second Experiment

In this case we have run the algorithms on instances associated to three groups of target functions. The first group consists of the following three polynomial functions. In all functions the variables take values in the range $[-100, 100]$.

$$F(x, y, z) = (x + y + z)^2 + 1 \qquad (9)$$

$$G(x,y,z) = \frac{1}{2}\,x + \frac{1}{3}\,y + \frac{2}{3}\,z \qquad (10)$$

$$K(x,y,z,w) = \frac{1}{2}\,x + \frac{1}{4}\,y + \frac{1}{6}\,z + \frac{1}{8}\,w \qquad (11)$$

The second group includes also three functions

$$g_1(x) = \begin{cases} 4x^2(3-4x) & x \in [0,0.5] \\ (4/3)x(4x^2 - 10x + 7) - 3/2 & x \in (0.5,0.75] \\ (16/3)x(x-1)^2 & x \in (0.75,1] \end{cases} \qquad (12)$$

$$g_2(x) = sin^2(2\pi x),\ x \in [0,1] \qquad (13)$$

$$g_3(x) = \frac{sin\sqrt{x_1^2 + x_2^2}}{x_1^2 + x_2^2},\ x_1,\,x_2 \in [-5,5] \qquad (14)$$

Finally, the third group of target functions is constituted by five unary functions of several classes: trigonometric functions, polynomial functions and one exponential function. These functions are the following

$$\begin{array}{ll} f_1(x) = x^4 + x^3 + x^2 + x & x \in [-15,15] \\ f_2(x) = e^{-sin\,3x + 2x} & x \in \left[-\frac{\pi}{2},\frac{\pi}{2}\right] \\ f_3(x) = e\,x^2 + \pi\,x & x \in [-\pi,\pi] \\ f_4(x) = cos(2x) & x \in [-\pi,\pi] \\ f_5(x) = min\{\frac{2}{x}, sin(x) + 1\} & x \in [0,15] \end{array} \qquad (15)$$

The parameters of the algorithms related with the population size, selection, recombination operators and termination condition at 250 generations, are the same as those fixed for the first experiment. Otherwise, for the problem instances associated to the functions of this second experiment, the upper bound for the length of the slp's is 16, the sample set has 30 points and the validation set is of size 200.

The corresponding sets of functions and terminals for each target function are showed in Table 4. In all cases the constants c_i are randomly generated in $[0,1]$. We have performed 100 executions of the three compared methods for each problem instance. The results presented in the following figures and tables are the same as those displayed for the first experiment.

Analyzing the empirical distribution of the executions, the VCD regularization method is clearly the best one for all target functions of the first and third group. There are very significative differences in the performance, specially for the functions of the first group, where the AIC and BIC methods found no good solutions in any of the 100 executions. However the three compared methods have a similar performance over the target functions of the second group. In fact, in the case of the bivariate function g_3 our VCD regularization seems to be worse than AIC and
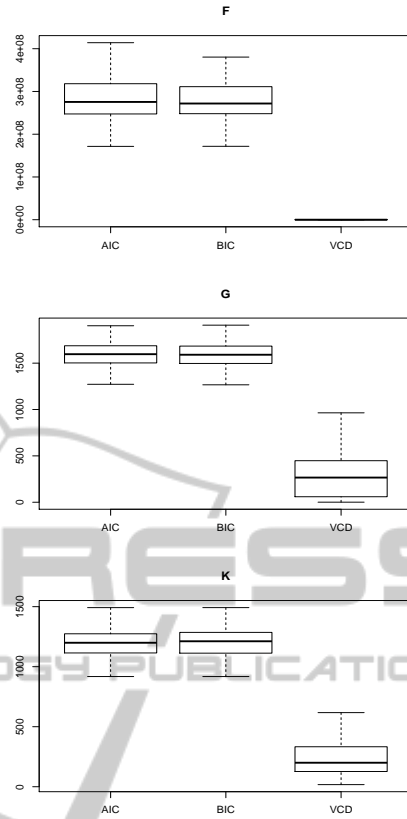

Figure 3: Empirical distribution of the executions. Second Experiment, first group of functions.

BIC. Considering that g_1 is a discontinuous piecewise polynomial function, g_2 is a *sine − square* function and g_3 is a two-dimensional *sin* function we could conjecture that for noisy problem instances associated with trigonometric or discontinuous target functions the outperformance of the VCD method is not so clear as it is for polynomial target functions. A similar situation occurs for the trigonometric functions of the third group f_4 and f_5. On the other hand, AIC and BIC methods always perform quite similar. This could be because both associated fitness functions (equations 6 and 7) take the same structure with very similar additive penalization terms.

The results displayed in Tables 5 and 6 agree with the above comments about the comparative study of the three methods. Note that VCD regularization obtains the lowest mean prediction risk for almost all the proposed target functions.

The crossed statistical hypothesis tests showed in Table 7 confirm VCD as the best method for the problem instances associated to the functions of the first and the third group.

Table 4: Function set and terminal set for the functions. $F = \{+, -, *, /\}$.

Function	Function set	Terminal set
F	F	$\{x, y, z\}$
G	F	$\{x, y, z, c_1, \ldots, c_6\}$
K	F	$\{x, y, z, w, c_1, \ldots, c_6\}$
g_1	$F \cup \{sign\}$	$\{x, c_1, \ldots, c_5\}$
g_2	$F \cup \{sin\}$	$\{x, c_1, \ldots, c_5\}$
g_3	$F \cup \{sqrt, sin\}$	$\{x_1, x_2, c_1, \ldots, c_5\}$
f_1	$F \cup \{sqrt\}$	$\{x\}$
f_2	$F \cup \{sqrt, sin, cos, exp\}$	$\{x, c_1\}$
f_3	$F \cup \{sin, cos\}$	$\{x, c_1\}$
f_4	$F \cup \{sqrt, sin\}$	$\{x, c_1\}$
f_5	$F \cup \{sin, cos\}$	$\{x, c_1\}$

Table 5: Values of means and variances: second experiment.

F	μ	σ	G	μ	σ
AIC	$2.83 \cdot 10^8$	$5.3 \cdot 10^7$	AIC	1627.1	236.87
BIC	$2.74 \cdot 10^8$	$4.12 \cdot 10^7$	BIC	1612.58	217.41
VCD	0	0	VCD	314.15	293.55
K	μ	σ	g_1	μ	σ
AIC	1199.14	107.03	AIC	0.15	0.06
BIC	1203.45	110.30	BIC	0.15	0.06
VCD	237.78	170.13	VCD	0.13	0.19
g_2	μ	σ	g_3	μ	σ
AIC	0.21	0.06	AIC	0.20	0.15
BIC	0.21	0.06	BIC	0.20	0.15
VCD	0.20	0.06	VCD	0.37	0.33
f_1	μ	σ	f_2	μ	σ
AIC	$7.6 \cdot 10^5$	$7.8 \cdot 10^5$	AIC	34.3	34.4
BIC	$7.5 \cdot 10^5$	$7.6 \cdot 10^5$	BIC	44.76	44.92
VCD	11.23	27.01	VCD	4.99	5.49
f_3	μ	σ	f_4	μ	σ
AIC	84.33	20	AIC	0.54	0.11
BIC	81.35	18.87	BIC	0.54	0.09
VCD	1.53	1.90	VCD	0.43	0.32
f_5	μ	σ			
AIC	0.49	0.08			
BIC	0.50	0.07			
VCD	0.36	0.19			

5.3 Third Experiment

We have executed the described evolutionary processes over some sample sets obtained from the KEEL-dataset repository (see (Alcalá-Fdez et al., 2011) [1]). KEEL-dataset includes sets of sample points for problems of different categories. We have selected three data sets corresponding to the following regression problems: *Abalone*, that consists in the prediction of the age of these molluscs from physical measurements; *Ailerons*, which addresses a control problem, flying a F16 aircraft and *AutoMPG8*, where the objective is to predict fuel consumption of a city cycle. The respective sizes of the data sets are 4177, 13750 and 392. We consider the same general configuration parameters and slp's of length bounded by 16. The set of functions consists of all the functions previously considered in this work and the terminal set contains the variables and a number of constants that equals the number of variables for each problem. That is 8 for *Abalone*, 40 for *Ailerons* and 7 for *AutoMPG8*.

[1] http://www.keel.es

Table 6: Prediction risk of the model obtained from the best execution: second experiment

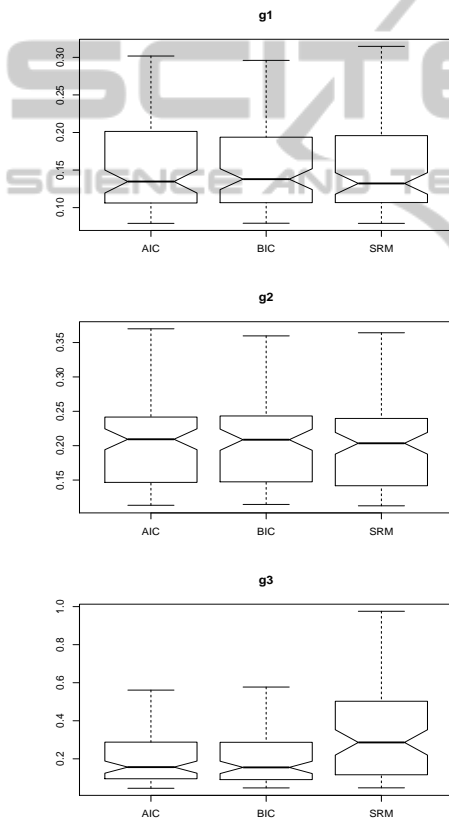| Problem instance	AIC	BIC	VCD
F	$1.7 \cdot 10^8$	$1.7 \cdot 10^8$	0
G	1272.03	1266.42	$0,69$
K	917.83	917.67	17.46
g_1	$7.91 \cdot 10^{-2}$	$7.93 \cdot 10^{-2}$	$7.91 \cdot 10^{-2}$
g_2	0.11	0.11	0.11
g_3	$4.53 \cdot 10^{-2}$	$4.71 \cdot 10^{-2}$	$4.75 \cdot 10^{-2}$
f_1	$3.02 \cdot 10^{-24}$	$3.24 \cdot 10^{-24}$	$2.26 \cdot 10^{-24}$
f_2	2.43	2.17	0.17
f_3	60.38	56.06	$2.77 \cdot 10^{-2}$
f_4	$5.5 \cdot 10^{-3}$	$1,1 \cdot 10^{-2}$	$9.9 \cdot 10^{-4}$
f_5	$0,15$	0.42	$5.03 \cdot 10^{-2}$

Figure 4: Empirical distribution of the executions. Second Experiment, second group of functions.

In all cases, and for each execution, we have divided the data sets into two equal parts: one of them was considered as the set of examples and the other was used as the validation set. Note that in this experiment the examples are not corrupted by noise. As usually, we have performed 100 executions for each problem instance and method.

Nevertheless the 40 variables that has Ailerons

problem, it seems to be the most easy one, as optimal solutions in almost all executions were obtained for the three methods. For the other two problems the studied methods also perform quite well. For these selected real problem instances the VCD regularization method is clearly better than the other two, as we can see considering the empirical distribution showed in figure 6. The results of the crossed statistical hypothesis tests with a significance value $\alpha = 0.05$ confirm the above affirmation.

6 CONCLUSIONS

Straight line programs constitute a promising structure for representing models in the Genetic Programming framework. Indeed, as it was published in a previous work, slp's outperform the traditional tree structure when GP strategies are applied for some kind of regression problems. In this paper we try to control the complexity of populations of slp's while they evolve in order to find good models for solving symbolic regression problem instances. The evolving structure is constructed from a set of functions that contains Pfaffian operators. We have considered the Vapnik Chervonenkis dimension as a complexity measure and we have found a theoretical upper bound of the VCD of families of slp's over Pfaffian operators as an important generalization of similar results for more simple sets of operators including rational functions. This theoretical upper bound is polynomial in the number of the non-scalar instructions of the family of the slp's. As a consequence of the main result, we propose a regularized fitness function included in a evolutionary strategy for solving symbolic regression problem instances. We have compared our fitness function based on the VCD upper bound with two well known statistical penalization criteria. The
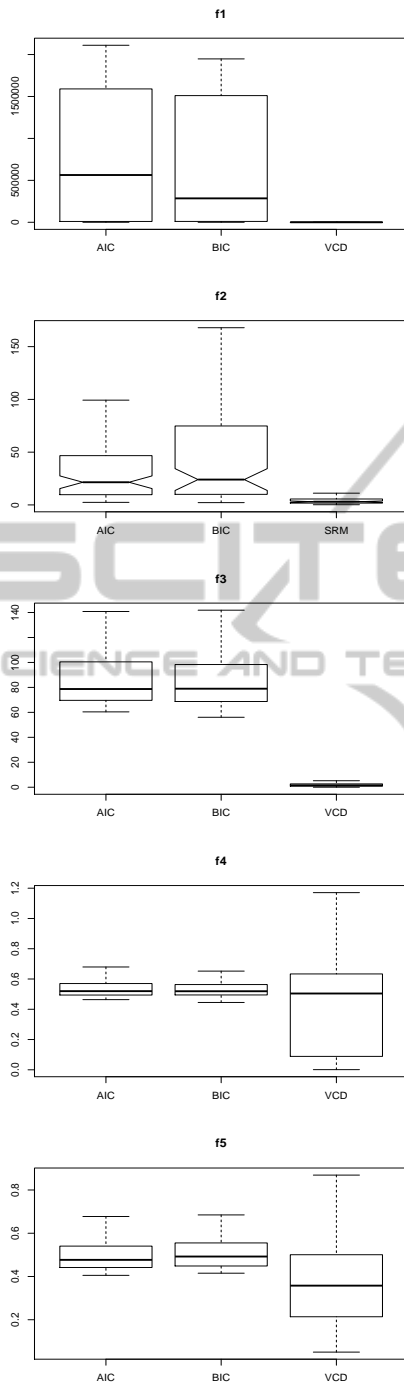
Figure 5: Empirical distribution of the executions. Second Experimente, third group of functions.

experimental results obtained after the execution of the compared strategies over three different groups of target functions, show that our proposed complexity measure and its corresponding penalization criterion is better than the others in almost all studied situa-

Table 7: Results of the crossed statistical hypothesis tests about the comparative quality of the studied strategies. Second experiment functions.

F	AIC	BIC	VCD
AIC		0.96	1
BIC	0.2		0.96
VCD	$3.7 \cdot 10^{-44}$	$3.7 \cdot 10^{-44}$	
G	AIC	BIC	VCD
AIC		0.91	1
BIC	0.70		1
VCD	$2.72 \cdot 10^{-43}$	$2.72 \cdot 10^{-43}$	
K	AIC	BIC	VCD
AIC		0.44	1
BIC	0.85		1
VCD	$2.72 \cdot 10^{-43}$	$2.72 \cdot 10^{-43}$	
g_1	AIC	BIC	VCD
AIC		0.66	0.79
BIC	0.75		0.87
VCD	0.63	0.75	
g_2	AIC	BIC	VCD
AIC		0.81	0.90
BIC	0.75		0.96
VCD	0.40	0.69	
g_3	AIC	BIC	VCD
AIC		0.84	$1.37 \cdot 10^{-3}$
BIC	0.76		$8.37 \cdot 10^{-4}$
VCD	1	1	
f_1	AIC	BIC	VCD
AIC		0.64	1
BIC	0.52		1
VCD	$1.24 \cdot 10^{-30}$	$2.39 \cdot 10^{-28}$	
f_2	AIC	BIC	VCD
AIC		0.19	1
BIC	0.79		1
VCD	$1.34 \cdot 10^{-15}$	$9 \cdot 10^{-19}$	
f_3	AIC	BIC	VCD
AIC		0.98	1
BIC	0.37		1
VCD	$1.08 \cdot 10^{-39}$	$1.07 \cdot 10^{-38}$	
f_4	AIC	BIC	VCD
AIC		0.86	$4.78 \cdot 10^{-2}$
BIC	0.43		$2.35 \cdot 10^{-2}$
VCD	$1.09 \cdot 10^{-5}$	$5.7 \cdot 10^{-6}$	
f_5	AIC	BIC	VCD
AIC		0.39	0.77
BIC	0.99		0.77
VCD	$4.72 \cdot 10^{-13}$	$6.5 \cdot 10^{-15}$	

tions, specially over the group of the three real problem instances, where the VCD regularization method is clearly the best.
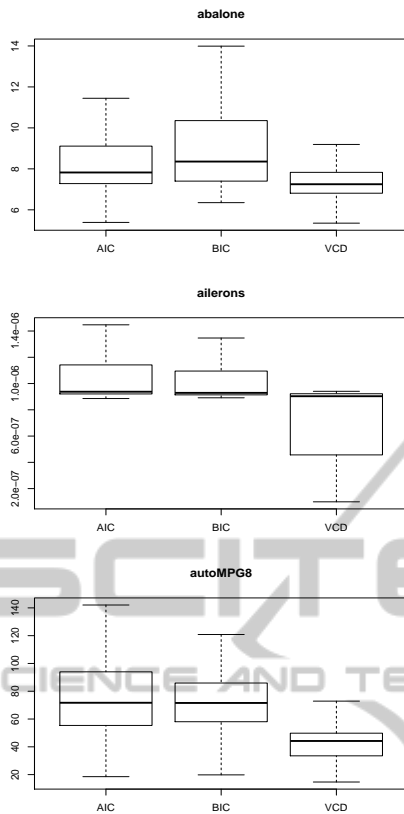
Figure 6: Empirical distribution of the executions, for functions associated to real problems.

Table 8: Prediction risk of the model obtained from the best execution.

Instance	AIC	BIC	VCD
Abalone	5.39	6.35	5.35
Ailerons	$1.22 \cdot 10^{-7}$	$1.01 \cdot 10^{-7}$	$9.93 \cdot 10^{-8}$
AutoMPG8	18.53	15.38	14.67

Table 9: Results of the crossed statistical hypothesis tests about the comparative quality of the studied strategies.

Abalone	AIC	BIC	VCD
AIC		$5.37 \cdot 10^{-2}$	1
BIC	0.95		1
VCD	$3.36 \cdot 10^{-6}$	$4.78 \cdot 10^{-9}$	
Ailerons	AIC	BIC	VCD
AIC		1	1
BIC	0.12		1
VCD	$9.26 \cdot 10^{-12}$	$5.3 \cdot 10^{-8}$	
AutoMPG8	AIC	BIC	VCD
AIC		0.76	1
BIC	0.99		1
VCD	$1.2 \cdot 10^{-2}$	$3.93 \cdot 10^{-4}$	

REFERENCES

Akaike, H. (1970). Statistical prediction information. *Ann. Inst. Statistic. Math*, 22:203–217.

Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., Garcia, S., Sánchez, L., and Herrera, F. (2011). Keel datamining sofware tool: Data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple–Valued Logic and Soft Computing*, 17(2–3):255–287.

Alonso, C. L., Montana, J. L., and Puente, J. (2008). Straight line programs: a new linear genetic programming approach. In *Proc. 20th IEEE International Conference on Tools with Artificial Intelligence (IC-TAI)*, pages 571–524.

Angluin, D. and Smith, C. (1983). Inductive inference: Theory and methods. *ACM Computing Surveys*, 15(3):237–269.

Berkowitz, S. (1984). On computing the determinant in small parallel time using a small number of processors. *Information Processing Letters*, 18:147–150.

Bernadro, J. and Smith, A. (1994). *Bayesian Theory*. John Willey & Sons.

Burguisser, P., Clausen, M., and Shokrollahi, M. (1997). *Algebraic Complexity Theory*. Springer.

Gabrielov, A. and Vorobjov, N. (2004). Complexity of computations with pfaffian and noetherian functions. In *Normal Forms, Bifurcations and Finiteness Problems in Differential Equations*. Kluwer.

Giusti, M., Heintz, J., Morais, J., Morgentern, J., and Pardo, L. (1997). Straight line programs in geometric elimination theory. *Journal of Pure and Applied Algebra*, 124:121–146.

Giusti, M. and Heinz, J. (1993). La détermination des points isolés et la dimension dúne varieté agebrique peut se faire en temps polynomial. In *Computational Algebraic Geometry and Commutative Algebra, Symposia Matematica XXXIV, ed. D. Eisenbud and L. Robbiano*, pages 216–256. Cambridge University Press.

Golberg, P. and Jerrum, M. (1995). Bounding the vapnik-chervonenkis dimension of concept classes parametrizes by real numbers. *Machine Learning*, 18(1):131–148.

Gori, M., Maggini, M., Martinelli, E., and Soda, G. (1998). Inductive inference from noisy examples using the hybrid finite state filter. *IEEE Transactions on Neural Networks*, 9(3):571–575.

Heintz, J., Roy, M., and Solerno, P. (1990). Sur la complexité du principe de tarski-seidenberg. *IBulletin de la Societé Mathematique de France*, 118:101–126.

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

Nikolaev, N. and Iba, H. (2001). Regularization approach to inductive genetic programming. *IEEE Transactions on Evolutionary Computation*, 5(4):359–375.

Oakley, H. (1994). Two scientific applications of genetic programming: Stack filters and nonlinear fitting to chaotic data. In *Advances in Genetic Programming*, pages 369–389. Cambridge, MA: MIT Press.

Poli, R. and Cagnoni, S. (1997). Evolution of pseudo-coloring algoritms for image enhancement with inter-active genetic programming. In *J.R. Koza, K.Deb, M. Dorigo, D.B. Fogel, M. Garzon, H. Iba and R.L. Riolo Eds*, pages 269–277. Cambridge, MA: MIT Press.

Shaoning, P. and Kasabov, N. (2004). Inductive vs trans-ductive inference, global vs local models: Svm, tsvm and svmt for gene expression classification problems. In *Proceedings IEEE International Joint Conference on Neural Networks*, pages 1197–1202.

Tackett, W. and Carmi, A. (1994). The donut problem: Scal-ability and generalization in genetic programming. In *Advances in Genetic Programming*, pages 143–176. Cambridge, MA: MIT Press.

Tenebaum, J., Griffiths, T., and Kemp, C. (2006). Theory based bayesian models of inductive learning and rea-soning. *Trends in Cognitive Sciences*, 10(7).

Vapnik, V. (1998). *Statistical Learning Theory*. John Willey & Sons.