COEVOLUTIONARY ARCHITECTURES WITH STRAIGHT LINE PROGRAMS FOR SOLVING THE SYMBOLIC REGRESSION PROBLEM

Cruz Enrique Borges, César L. Alonso*, José Luis Montaña
Departamento de Matemáticas, Estadística y Computación, Universidad de Cantabria, 39005 Santander, Spain
**Centro de Inteligencia Artificial, Universidad de Oviedo, Campus de Viesques, 33271 Gijón, Spain*

Marina de la Cruz Echeandía, Alfonso Ortega de la Puente
Departamento de Ingeniería Informática, Escuela Politécnica Superior, Universidad Autónoma de Madrid, Madrid, Spain

Keywords: Genetic programming, Straight-line programs, Coevolution, Symbolic regression.

Abstract: To successfully apply evolutionary algorithms to the solution of increasingly complex problems we must develop effective techniques for evolving solutions in the form of interacting coadapted subcomponents. In this paper we present an architecture which involves cooperative coevolution of two subcomponents: a genetic program and an evolution strategy. As main difference with work previously done, our genetic program evolves straight line programs representing functional expressions, instead of tree structures. The evolution strategy searches for good values for the numerical terminal symbols used by those expressions. Experimentation has been performed over symbolic regression problem instances and the obtained results have been compared with those obtained by means of Genetic Programming strategies without coevolution. The results show that our coevolutionary architecture with straight line programs is capable to obtain better quality individuals than traditional genetic programming using the same amount of computational effort.

1 INTRODUCTION

Coevolutionary strategies can be considered as an interesting extension of the traditional evolutionary algorithms. Basically, coevolution involves two or more evolutionary processes with interactive performance. Initial ideas on modelling coevolutionary processes were formulated in (Maynard, 1982), (Axelrod, 1984) or (Hillis, 1991). A coevolutionary strategy consists in the evolution of separate populations using their own evolutionary parameters (i.e. genotype of the individuals, recombination operators, ...) but with some kind of interaction between these populations. Two basic classes of coevolutionary algorithms have been developed: competitive algorithms and cooperative algorithms. In the first class, the fitness of an individual is determined by a series of competitions with other individuals. Competition takes place between the partial evolutionary processes coevolving and the success of one implies the failure of the other (see, for example, (Rosin and Belew, 1996)). On the other hand, in the second class the fitness of an individual is determined by a series of collaborations with other individuals from other populations.

The standard approach of cooperative coevolution is based on the decomposition of the problem into several partial components. The structure of each component is assigned to a different population. Then the populations are evolved in isolation from one another but in order to compute the fitness of an individual from a population, a set of collaborators are selected from the other populations. Finally a solution of the problem is constructed by means of the combination of partial solutions obtained from the different populations. Some examples of application of cooperative coevolutionary strategies for solving problems can be found in (Wiegand et al., 2001) and (Casillas et al., 2006).

This paper focuses on the design and the study of several coevolutionary strategies between Genetic Programming (GP) and Evolutionary Algorithms (EA). Although in the cooperative systems the coevolving populations usually are homogeneous (i.e. with similar genotype representations), in this case

Enrique Borges C., L. Alonso C., Luis Montaña J., de la Cruz Echeandia M. and Ortega de la Puente A..
COEVOLUTIONARY ARCHITECTURES WITH STRAIGHT LINE PROGRAMS FOR SOLVING THE SYMBOLIC REGRESSION PROBLEM.
DOI: 10.5220/0003075100410050
In *Proceedings of the International Conference on Evolutionary Computation* (ICEC-2010), pages 41-50
ISBN: 978-989-8425-31-7

we deal with two heterogeneous populations: one composed by elements of a structure named *straight line program* (slp) that represents programs and the other one composed by vectors of real constants. The coevolution between GP and EA was applied with promising results in (Vanneschi et al., 2001). In that case a population of trees and another one of fixed length strings were used.

We have applied the strategies to solve symbolic regression problem instances. The problem of symbolic regression consists in finding in symbolic form a function that fits a given finite sample set of data points. More formally, we consider an input space $X = \mathbb{R}^n$ and an output space $Y = \mathbb{R}$. We are given a set of m pairs sample $z = (x_i, y_i)_{1 \leq i \leq m}$. The goal is to construct a function $f : X \rightarrow Y$ which predicts the value $y \in Y$ from a given $x \in X$. The empirical error of a function f with respect to z is:

$$\varepsilon_z(f) = \frac{1}{m} \sum_{i=1}^{m} (f(x_i) - y_i)^2 \qquad (1)$$

which is known as the mean square error (MSE).

In our coevolutionary processes for finding the function f, the GP will try to guess the shape of the function whereas the EA will try to adjust the coefficients of the function. The motivation is to exploit the following intuitive idea: once the shape of the symbolic expression representing some optimal function has been found, we try to determine the best values of the coefficients appearing in the symbolic expression. One simple way to exemplify this situation is the following. Assume that we have to guess the equation of a geometric figure. If somebody (for example a GP algorithm) tells us that this figure is a quartic function, it only remains for us to guess the appropriate coefficients. This point of view is not new and it constitutes the underlying idea of many successful methods in Machine Learning that combine a space of hypotheses with least square methods. Previous work in which constants of a symbolic expression have been effectively optimized has also dealt with memetic algorithms, in which classical local optimization techniques as gradient descent (Topchy and Punch, 2001), linear scaling (Keijzer, 2003) or other methods based on diversity measures (Ryan and Keijzer, 2003) were used.

The paper is organized as follows: section 2 provides the definition of the structure that will represent the programs and also includes the details of the designed GP algorithm. In section 3 we describe the EA for obtaining good values for the constants. Section 4 presents the cooperative coevolutionary architecture used for solving symbolic regression problem instances. In section 5 an experimental comparative study of the performance of our coevolutionary strategies is done. Finally, section 6 draws some conclusions and addresses future research directions.

2 GP WITH STRAIGHT LINE PROGRAMS

In the GP paradigm, the evolved computer programs are usually represented by directed trees with ordered branches (Koza, 1992). We use in this paper a structure for representing programs called *straight line program* (slp). A slp consists of a finite sequence of computational assignments where each assignment is obtained by applying some function to a set of arguments that can be variables, constants or pre-computed results. The slp structure can describe complex computable functions using less amount of computational resources than GP-trees, as they can reuse previously computed results during the evaluation process. Now follows the formal definition of this structure.

Definition. Let $F = \{f_1, \ldots, f_n\}$ be a set of functions, where each f_i has arity a_i, $1 \leq i \leq n$, 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 for each $k \in \{1, \ldots, l\}$, $I_k \equiv u_k := f_{j_k}(\alpha_1, \ldots, \alpha_{a_{j_k}})$; *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 $1 < k \leq l$.

The set of terminals T satisfies $T = V \cup C$ where $V = \{x_1, \ldots, x_p\}$ 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.

Observe that a slp $\Gamma = \{I_1, \ldots, I_l\}$ is identified with the set of variables u_i that are introduced by means of the instructions I_i. Thus the slp Γ can be denoted by $\Gamma = \{u_1, \ldots, u_l\}$. Each of the non-terminal variables u_i represents an expression over the set of terminals T constructed by a sequence of recursive compositions from the set of functions F.

An output set of a slp $\Gamma = \{u_1, \ldots, u_l\}$ is any set of non-terminal variables of Γ, that is $O(\Gamma) = \{u_{i_1}, \ldots, u_{i_t}\}$. Provided that $V = \{x_1, \ldots, x_p\} \subset T$ is the set of terminal variables, the function computed by Γ, denoted by $\Phi_\Gamma : I^p \rightarrow O^t$, 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 substitute the variable x_k by a_k; $1 \leq k \leq p$.

Example. Let F be the set given by the three binary standard arithmetic operations, $F = \{+, -, *\}$ and let $T = \{1, x_1, x_2\}$ be the set of terminals. In this situation

any slp over F and T is a finite sequence of instructions where each instruction represents a polynomial in two variables with integer coefficients. If we consider the following slp Γ of length 5 with output set $O(\Gamma) = \{u_5\}$:

$$\Gamma \equiv \begin{cases} u_1 := x_1 + 1 \\ u_2 := u_1 * u_1 \\ u_3 := x_2 + x_2 \\ u_4 := u_2 * u_3 \\ u_5 := u_4 - u_3 \end{cases} \quad (2)$$

the function computed by Γ is the polynomial

$$\Phi_\Gamma = 2x_2(x_1+1)^2 - 2x_2$$

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 underling computation model in Algebraic Complexity Theory (Burguisser et al., 1997). They have been used in linear algebra problems (Berkowitz, 1984), in quantifier elimination (Heintz et al., 1990) and in algebraic geometry (Giusti et al., 1997). Recently, slp's have been presented as a promising alternative to the trees in the field of Genetic Programming, with a good performance in solving some regression problem instances (Alonso et al., 2008). A slp $\Gamma = \{u_1, \ldots, u_l\}$ over F and T with output set $O(\Gamma) = \{u_l\}$ could also be considered as a grammar with $T \cup F$ as the set of terminals, $\{u_1, \ldots, u_l\}$ as the set of variables, u_l the start variable and the instructions of Γ as the rules. This grammar only generates one word that is the expression represented by the slp Γ. Note that this is not exactly Grammar Evolution (GE). In GE there is a user specified grammar and the individuals are integer strings which code for selecting rules from the provided grammar. In our case each individual is a context-free grammar generating a context-free language of size one.

Hence we will work with slp's over a set F of functions and a set T of terminals. The elements of T that are constants, i.e. $C = \{c_1, \ldots, c_q\}$, they are not fixed numeric values but references to numeric values. Hence, specializing each c_i to a fixed value we obtain a specific slp whose corresponding semantic function is a candidate solution for the problem instance.

For constructing each individual Γ of the initial population, we adopt the following process: for each instruction $u_k \in \Gamma$ first an element $f \in F$ is randomly selected and then the function arguments of f are also randomly chosen in $T \cup \{u_1, \ldots, u_{k-1}\}$ if $k > 1$ and in T if $k = 1$. We will consider populations with individuals of equal length L, where L is selected by the user. In this sense, note that given a slp $\Gamma = \{u_1, \ldots, u_l\}$ and $L \geq l$, we can construct the slp $\Gamma' = \{u_1, \ldots, u_{l-1}, u'_l, \ldots, u'_{L-1}, u'_L\}$, where $u'_L = u_l$ and u'_k, for $k \in \{l, \ldots, L-1\}$, is any instruction. If we consider the same output set for Γ and Γ' is easy to see that they represent the same function, i.e. $\Phi_\Gamma \equiv \Phi_{\Gamma'}$.

Given a symbolic regression problem instance with a sample set $z = (x_i, y_i) \in \mathbb{R}^n \times \mathbb{R}$, $1 \leq i \leq m$, and let Γ be a specific slp over F and T obtained by means of the specialization of the constant references $C = \{c_1, \ldots, c_q\}$. In this situation, the fitness of Γ is defined by the following expression:

$$\mathcal{F}_z(\Gamma) = \varepsilon_z(\Phi_\Gamma) = \frac{1}{m}\sum_{i=1}^{m}(\Phi_\Gamma(x_i) - y_i)^2 \quad (3)$$

That is, the fitness is the empirical error of the function computed by Γ, with respect to the sample set of data points z.

We will use the following recombination operators for the slp structure.

slp-crossover. Let $\Gamma = \{u_1, \ldots, u_L\}$ and $\Gamma' = \{u'_1, \ldots, u'_L\}$ be two slp's over F and T. For the construction of an offspring, first a position k in Γ is randomly selected; $1 \leq k \leq L$. Let $S_{u_k} = \{u_{j_1}, \ldots, u_{j_m}\}$ be the set of instructions of Γ involved in the evaluation of u_k. Assume that $j_1 < \ldots < j_m$. Next we randomly select a position t in Γ' with $m \leq t \leq L$ and we substitute in Γ' the subset of instructions $\{u'_{t-m+1}, \ldots, u'_t\}$ by the instructions of Γ in S_{u_k} suitably renamed. The renaming function $\mathcal{R}$ applied to the elements of S_{u_k} is defined as $\mathcal{R}(u_{j_i}) = u'_{t-m+i}$, for all $i \in \{1, \ldots, m\}$. With this process we obtain the first offspring of the crossover operation. For the second offspring we analogously repeat this strategy, but now selecting first a position k' in Γ'.

The underlying idea of the slp-crossover is to interchange subexpressions between Γ and Γ'. The following example illustrates this fact.

Example. Let us consider two slp's:

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

If $k = 3$ then $S_{u_3} = \{u_1, u_3\}$ (in bold font). t must be selected in $\{2, \ldots, 5\}$. Assumed that $t = 3$, then the first offspring is:

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

that contains the subexpression of Γ represented by u_3, and the rest of its instructions are taked from Γ'.

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

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

Mutation. When mutation is applied to a slp Γ, the first step consists in selecting an instruction $u_i \in \Gamma$ at random. Then a new random selection is made within the arguments of the function $f \in F$ that appears in the instruction u_i. Finally the selected argument is substituted by another one in $T \cup \{u_1, \ldots, u_{i-1}\}$ randomly chosen.

As it is well known, the reproduction operation applied to an individual returns an exact copy of the individual.

3 THE EA TO ADJUST THE CONSTANTS

In this section we describe an EA that provides good values for the numeric terminal symbols $C = \{c_1, \ldots, c_q\}$ appearing in the populations of slp's that evolve during the GP process. Assume a population $P = \{\Gamma_1, \ldots, \Gamma_N\}$ constituted by N slp's over F and $T = V \cup C$. Let $[a,b] \subset \mathbb{R}$ be the search space for the constants c_i, $1 \leq i \leq q$. In this situation, an individual $\overline{c}$ is represented by a vector of floating point numbers in $[a,b]^q$.

There are several ways of defining the fitness of a vector of constants $\overline{c}$, but in all of them the current population P of slp's that evolves in the GP process is needed. So, given a sample set $z = (x_i, y_i) \in \mathbb{R}^n \times \mathbb{R}$, $1 \leq i \leq m$, that defines a symbolic regression instance, and given a vector of values for the constants $\overline{c} = (c_1, \ldots, c_q)$, we could define the fitness of $\overline{c}$ with the following expression:

$$\mathcal{F}_z^{EA}(\overline{c}) = \min\{\mathcal{F}_z(\Gamma_i^{\overline{c}}); 1 \leq i \leq N\} \qquad (4)$$

where $\mathcal{F}_z(\Gamma_i^{\overline{c}})$ is computed by equation 3 and represents the fitness of the slp Γ_i after the specialization of the references in C to the corresponding real values of $\overline{c}$.

Observe that when the fitness of $\overline{c}$ is computed by means of the above formula, the GP fitness values of a whole population of slp's are also computed. This could be a lot of computational effort when the size of both populations increases. In order to prevent the above situation new fitness functions for $\overline{c}$ can be considered, where only a subset of the population P of the slp's is evaluated. Previous work in cooperative coevolutionary architectures suggests two basic methods for selecting the subset of collaborators (Wiegand et al., 2001): The first one in our case consists in the selection of the best slp of the current population P corresponding the GP process. The second one selects two individuals from P: the best one and a random slp.

Crossover. We will use arithmetic crossover (Schwefel, 1981). Thus, in our EA, the crossover of two individuals $\overline{c}_1$ and $\overline{c}_2 \in [a,b]^q$ produces two offsprings $\overline{c}'_1$ and $\overline{c}'_2$ which are linear combinations of their parents.

$$\overline{c}'_1 = \lambda \cdot \overline{c}_1 + (1-\lambda) \cdot \overline{c}_2; \quad \overline{c}'_2 = \lambda \cdot \overline{c}_2 + (1-\lambda) \cdot \overline{c}_1 \quad (5)$$

In our implementation we randomly choose $\lambda \in (0,1)$ for each crossover operation.

Mutation. A non-uniform mutation operator adapted to our search space $[a,b]^q$, which is convex, is used (Michalewicz et al., 1994). The following expressions define our mutation operator, with $p = 0.5$.

$$\overline{c}_k^{t+1} = \overline{c}_k^t + \Delta(t, b - \overline{c}_k^t), \text{ with probability } p \qquad (6)$$

and

$$\overline{c}_k^{t+1} = \overline{c}_k^t - \Delta(t, \overline{c}_k^t - a), \text{ with probability } 1-p \quad (7)$$

$k = 1, \ldots, q$ and t is the current generation. The function Δ is defined as $\Delta(t,y) = y \cdot r \cdot (1 - \frac{t}{T})$ where r is a random number in [0,1] and T represents the maximum number of generations. Note that function $\Delta(t,y)$ returns a value in $[0,y]$ such that the probability of obtaining a value of $\Delta(t,y)$ close to zero increases as t increases. Hence the mutation operator searchs the space uniformly initially (when t is small), and very locally at later stages.

In our EA we will use q-tournament as the selection procedure.

4 THE COEVOLUTIONARY ARCHITECTURE

In our case the EA for tuning the constants is subordinated to the main GP process with the slp's. Hence, several collaborators are used during the computation of the fitness of a vector of constants $\overline{c}$, whereas only the best vector of constants is used to compute the fitness of a population of slp's.

A basic cooperative coevolutionary strategy begins with the initialization of both populations. First the fitness of the individuals of the slp's population are computed considering a randomly selected vector of constants as collaborator. Then alternative generations of both cooperative algorithms in a round-robin

fashion are performed. This strategy can be generalized in a natural way by the execution of alternative turns of both algorithms. We will consider a *turn* as the isolated and uninterrupted evolution of one population for a fixed number of generations. Note that if a turn consists of only one generation we obtain the basic strategy. We display below the algorithm describing this cooperative coevolutionary architecture:

```
begin
Pop_slp := initialize_GP-slp_population
Pop_const := initialize_EA-constants_population
Const_collabor := random(Pop_const)
evaluate(Pop_slp,Const_collabor)
While (not termination condition) do
  Pop_slp := turn_GP(Pop_slp,Const_collabor)
  Collabor_slp := {best(Pop_slp),
                   random(Pop_slp)}
  Pop_const := turn_EA(Pop_const,Collabor_slp)
  Const_collabor := best(Pop_const)
end
```

5 EXPERIMENTATION

5.1 Experimental Settings

The experimentation consists in the execution of the proposed cooperative coevolutionary strategies, considering several types of target functions. Two experiments were performed.

For the first experiment two groups of target functions are considered: the first group includes 100 randomly generated univariate polynomials whose degrees are bounded by 5 and the second group consists of 100 target functions represented by randomly generated slp's over $F = \{+,-,*,/,sqrt,sin,cos,ln,exp\}$ and $T = \{x,c\}, c \in [-1,1]$, with length 16. We will name this second group "target slp's".

A second experiment is also performed solving symbolic regression problem instances associated to the following three multivariate functions:

$$f_1(x,y,z) = (x+y+z)^2+1 \tag{8}$$

$$f_2(x,y,z,w) = \frac{1}{2}x+\frac{1}{4}y+\frac{1}{6}z+\frac{1}{8}w \tag{9}$$

$$f_3(x,y) = x\,y+sin((x-1)\,(y+1)) \tag{10}$$

For every execution the sample set is constituted by 30 points. In the case of the functions that belong to the first experiment, the sample points are in the range $[-1,1]$. For the functions f_1 and f_2 the points are in the range $[-100,100]$ for all variables. Finally function f_3 varies in the range $[-3,3]$ along each axis.

The individuals are slp's over $F = \{+,-,*,/,sqrt\}$ in the executions related to the 100 generated polynomials and to the functions f_1 and f_2. The function set F is incremented with the operation *sin* for the problem instance associated to f_3 and also with the operations *cos*, *ln* and *exp* for the group of target slp's.

Besides the variables, the terminal set also includes two references to constants for the polynomials and only one reference to a constant for the rest of the target functions. The constants take values in $[-1,1]$.

The particular settings for the parameters of the GP process are the following: population size: 200, crossover rate: 0.9, mutation rate: 0.05, reproduction rate: 0.05, 5-tournament as selection procedure and maximum length of the slp's: 16. In the case of the EA that adjusts the constants, the population includes 100 vector of constants, crossover rate: 0.9, mutation rate: 0.1 and 2-tournament as selection procedure. For all the coevolutionary strategies, the computation of the fitness of an slp during the GP process will use the best vector of constants as collaborator, whereas in order to compute the fitness of a vector of constants in the EA process, we consider a collaborator set containing the best slp of the population and another one randomly selected. Both processes are elitist and a generational replacement between populations is used. But in the construction of the new population, the offsprings generated 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. Our motivation is to prevent premature convergence and to maintain diversity in the population.

We compare the standard GP-slp strategy without coevolution with three coevolutionary strategies that follow the general architecture described in section 4. We have implemented and executed over the same sets of problem instances, the well known GP-tree strategy with the standard recombination operators. Thus the obtained results are included in the comparative study.

The first coevolutionary strategy, named Basic GP-EA (BGPEA), consists in the alternative execution of one generation of each cooperative algorithm. The second strategy, named Turns GP-EA (TGPEA), generalizes BGPEA by means of the execution of alternative turns of each algorithm. Finally, the third strategy executes first a large turn of the GP algorithm with the slp's and then follows the execution of the EA related with the constants until termination condition was reached. This strategy is named Separated GP-EA (SGPEA). In the case of TGPEA we have considered a GP turn as the evolution of the population of slp's during 25 generations. On the other hand, an EA turn consists of 5 generations in the evo-

lution of the population related to the constants. In SGPEA strategy we divide the computational effort between the two algorithms: 90% for GP and 10% for EA. The computational effort (CE) is defined as the total number of basic operations that have been computed up to that moment.

In the first experiment one execution for each strategy has been performed over the 200 generated target functions. On the other hand, in the second experiment we have executed all strategies 100 times for each of the three multivariate functions f_1, f_2 and f_3. For all the executions the evolution finished after 10^7 basic operations have been computed.

5.2 Experimental Results

Frequently, when different Genetic Programming strategies for solving symbolic regression instances are compared, the quality of the final selected model is evaluated by means of its corresponding fitness value over the sample set. But with this quality measure it is not possible to distinguish between good executions and overfiting executions. Then it makes sense to consider another new set of unseen points, called the *validation set*, in order to give a more appropriate indicator of the quality of the selected model. So, let $(x_i,y_i)_{1\leq i\leq n_{test}}$ 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 sample set. Then the *validation fitness* $vf_{n_{test}}$ is defined by the mean square error (MSE) between the values of f and the true values of the target function g over the validation set:

$$vf_{n_{test}} = \frac{1}{n_{test}} \sum_{i=1}^{n_{test}} (f(x_i) - y_i)^2 \qquad (11)$$

An execution will be considered successful if the final selected model f has validation fitness less than 10% of the range of the sample set $z = (x_i,y_i)_{1\leq i\leq 30}$. That is:

$$vf_{n_{test}} \leq 0.1 \left| \max_{1\leq i\leq 30} y_i - \min_{1\leq i\leq 30} y_i \right| \qquad (12)$$

On the other hand, an execution will be spurious if the validation fitness of the selected model verifies:

$$vf_{n_{test}} \geq 1.5|Q_3 - Q_1| \qquad (13)$$

Were Q_1 and Q_3 represent, respectively, the first and third quartile of the empirical distribution of the executions in terms of the validation fitness. The spurious executions will be removed from the experiment.

In what follows we shall present a complete statistical comparative study about the performance of the described coevolutionary strategies. For both experiments we will show the empirical distribution of the non-spurious executions as well as the values of the mean, variance, median, worst and best execution in terms of the validation fitness. We also present statistical hypothesis tests in order to determine if some strategy is better than the others. We consider a validation set of 200 new and unseen points randomly generated.

Table 1: Spurious and success rates for each strategy and group of target functions.

$P_5^{\mathcal{R}}[X]$	spurious	success
$SGPEA$	13%	100%
$TGPEA$	8%	99%
$BGPEA$	11%	100%
$GP-slp$	12%	100%
$GP-tree$	12%	100%
$SLP(F,T)$	spurious	success
$SGPEA$	16%	98%
$TGPEA$	13%	99%
$BGPEA$	13%	98%
$GP-slp$	14%	96%
$GP-tree$	14%	99%

5.2.1 Experiment 1

We shall denote the polynomial set as $P_5^{\mathcal{R}}[X]$ and the set of target slp's over F and T as $SLP(F,T)$. Table 1 displays for each strategy the spurious and success rates of the executions. Note that the success rate is computed after removing the spurious executions. Figure 1 presents the empirical distribution of the executions over the two groups of generated target functions. This empirical distribution is displayed using standard box plot notation with marks at best execution, 25%, 50%, 75% and worst execution, considering the validation fitness of the selected model. Table 2 specifies the values of the validation fitness for the worst, median and best execution. Finally Table 3 shows the means an variances. Note that for these two groups of functions one execution per target function was performed.

Analyzing the information given by the above tables and figure we could deduce the following facts:

1. All methods have a similar rate of spurious executions and also the success rate is almost equal for all strategies.
2. The empirical distributions of the non-spurious runs are again very similar for all the studied strategies. Probably for the group of polynomials, SGPEA is slightly better than the others. Observe in figure 1 that this strategy has the corresponding box smaller and a little below than the other methods. SGPEA also has the best mean and variance

Table 2: Minimal, median and maximal values of the validation fitness for each method and group of target functions.

$P_5^{\mathcal{R}}[X]$	min	med	max
$SGPEA$	$3.25 \cdot 10^{-3}$	$3.33 \cdot 10^{-2}$	0.17
$TGPEA$	$2.27 \cdot 10^{-3}$	$4.58 \cdot 10^{-2}$	0.2
$BGPEA$	$1.25 \cdot 10^{-3}$	$3.58 \cdot 10^{-2}$	0.23
$GP-slp$	$2.26 \cdot 10^{-3}$	$4.48 \cdot 10^{-2}$	0.17
$GP-tree$	$4.64 \cdot 10^{-3}$	$4.42 \cdot 10^{-2}$	0.19
$SLP(F,T)$	min	med	max
$SGPEA$	0	$1.6 \cdot 10^{-3}$	0.12
$TGPEA$	0	$1.66 \cdot 10^{-3}$	$7.69 \cdot 10^{-2}$
$BGPEA$	0	$2.84 \cdot 10^{-3}$	0.13
$GP-slp$	0	$1.61 \cdot 10^{-3}$	$9.32 \cdot 10^{-2}$
$GP-tree$	0	$4.29 \cdot 10^{-3}$	0.1

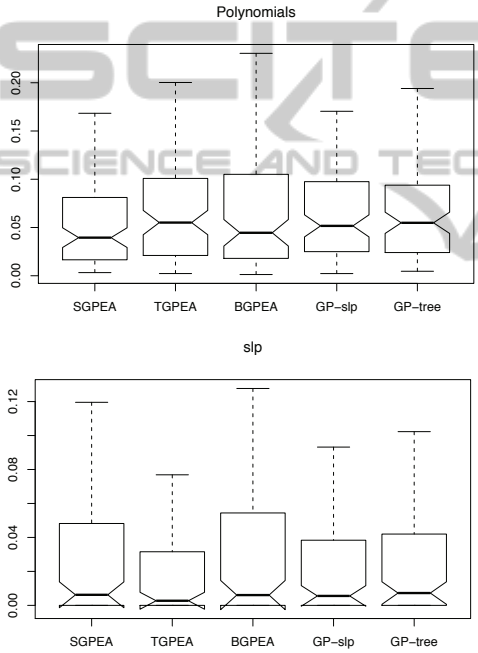

Figure 1: Empirical distributions of the non-spurious executions for both groups of functions and for each strategy.

values (table 3). Nevertheless for the group of target slp's it seems that the best method is TGPEA.

3. All strategies perform quite well over the target functions of this experiment, specially for the group of target slp's. Considering the values of mean and variance and the fact that for each function only one execution was performed, probably the BGPEA method is a little worse than the others.

With the objective of justify the comparative quality of the studied strategies we have made statistical hypothesis tests between them, which results are showed

Table 3: Values of means and variances.

$P_5^{\mathcal{R}}[X]$	μ	σ
$SGPEA$	$4.39 \cdot 10^{-2}$	$3.83 \cdot 10^{-2}$
$TGPEA$	$6.03 \cdot 10^{-2}$	$5.23 \cdot 10^{-2}$
$BGPEA$	$5.49 \cdot 10^{-2}$	$5.22 \cdot 10^{-2}$
$GP-slp$	$5.19 \cdot 10^{-2}$	$3.82 \cdot 10^{-2}$
$GP-tree$	$5.62 \cdot 10^{-2}$	$4.26 \cdot 10^{-2}$
$SLP(F,T)$	μ	σ
$SGPEA$	$1.62 \cdot 10^{-2}$	$2.79 \cdot 10^{-2}$
$TGPEA$	$1.14 \cdot 10^{-2}$	$1.78 \cdot 10^{-2}$
$BGPEA$	$1.96 \cdot 10^{-2}$	$3 \cdot 10^{-2}$
$GP-slp$	$1.49 \cdot 10^{-2}$	$2.33 \cdot 10^{-2}$
$GP-tree$	$1.5 \cdot 10^{-2}$	$2.23 \cdot 10^{-2}$

in table 4. Roughly speaking, the null-hypothesis in each test with associated pair (i, j) is that strategy i is not better than strategy j. Hence if value a_{ij} of the element (i, j) in table 4 is less than a significance value α, we can reject the corresponding null-hypothesis.

From the results presented in table 4 and with significance values of α between 0.05 and 0.1, we can conclude that for the group of polynomials the best strategy is SGPEA whereas there is no clear winner strategy for the group of target slp's.

5.2.2 Experiment 2

In this experiment, the multivariate functions described by the expressions 8, 9 and 10 have been considered as target functions. We have performed 100 executions for each strategy and function. In the following tables and figures we present for the new functions the same results as those presented for the target functions of experiment 1.

In terms of success rates, the coevolutionary methods are of similar performance and they seem to be better than the standard GP-slp strategy without coevolution. Nevertheless we can see that all the strategies outperform the standard GP-tree procedure. This fact is more clear after observing in figure 2 the empirical distributions of the non-spurious executions. For the multivariate polynomial of degree two, f_1, the best strategy is SGPEA whereas for the linear polynomial f_2 with four variables, TGPEA seems to be better than the other strategies. In the case of the trigonometric function f_3 it is not clear which is the best method. But in any case, the standard GP-tree method is the worst of the studied strategies.

In table 6 it can be seen that the validation fitness has a big range, specially for functions f_1 and f_2. Hence, the mean and variance, displayed in table 7, have also big values for the above two target functions. Indeed the values are very big for f_1, al-

Table 4: Results of the crossed statistical hypothesis tests about the comparative quality of the studied strategies.

$P_5^{\mathcal{R}}[X]$	$SGPEA$	$TGPEA$	$BGPEA$	$GP-slp$	$GP-tree$
$SGPEA$	1	$2.51 \cdot 10^{-2}$	0.27	$8.9 \cdot 10^{-2}$	$6.1 \cdot 10^{-2}$
$TGPEA$	0.78	1	0.8	0.53	0.45
$BGPEA$	0.9	0.3	1	0.12	0.17
$GP-slp$	0.99	0.28	0.59	1	0.4
$GP-tree$	1	0.61	0.77	0.75	1
$SLP(F,T)$	$SGPEA$	$TGPEA$	$BGPEA$	$GP-slp$	$GP-tree$
$SGPEA$	1	0.82	0.44	0.82	0.39
$TGPEA$	0.54	1	0.14	0.44	0.28
$BGPEA$	0.99	1	1	0.93	0.7
$GP-slp$	0.74	0.93	0.49	1	0.39
$GP-tree$	0.82	0.99	0.49	0.9	1

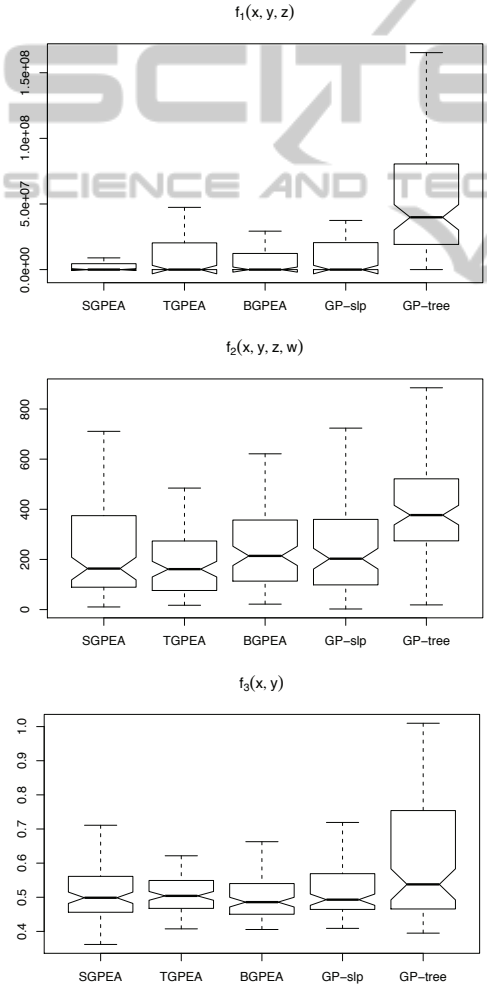

Figure 2: Empirical distributions of the non-spurious executions.

though the best execution over this function has validation fitness equal to zero for all methods. Note that for the above two functions the points are in the range $[-100,100]$ for all variables. Considering the results that appear in these two tables, we could say that the coevolutionary strategies outperform the non-coevolutionary ones.

Table 5: Spurious and success rates of the executions over the multivariate functions.

$f_1(x,y,z)$	spurious	success
$SGPEA$	25%	74%
$TGPEA$	13%	69%
$BGPEA$	19%	66%
$GP-slp$	11%	62%
$GP-tree$	5%	12%
$f_2(x,y,z,w)$	spurious	success
$SGPEA$	7%	42%
$TGPEA$	7%	49%
$BGPEA$	6%	41%
$GP-slp$	7%	39%
$GP-tree$	12%	14%
$f_3(x,y)$	spurious	success
$SGPEA$	8%	100%
$TGPEA$	14%	100%
$BGPEA$	9%	100%
$GP-slp$	9%	100%
$GP-tree$	20%	100%

Finally, as it was done in experiment 1, we have made for the three functions the crossed statistical hypothesis tests between all pairs of the considered strategies and the results are showed in table 8. Considering a significant value $\alpha = 0.05$ it seems that for the function f_1 SGPEA is the best strategy but there is no clear winner coevolutionary strategy for the functions f_2 and f_3. Nevertheless such results confirm that the coevolutionary methods are promising strategies for solving symbolic regression problem instances and that the slp structure is clearly better for representing the models than the tree structure.

Table 8: Results of the crossed statistical hypothesis tests.

$f_1(x,y,z)$	$SGPEA$	$TGPEA$	$BGPEA$	$GP-slp$	$GP-tree$
$SGPEA$	1	$3.39\cdot10^{-4}$	$1.41\cdot10^{-3}$	$9.23\cdot10^{-5}$	$4.53\cdot10^{-31}$
$TGPEA$	1	1	1	0.44	$7.13\cdot10^{-19}$
$BGPEA$	0.97	0.12	1	$2.66\cdot10^{-2}$	$7.46\cdot10^{-24}$
$GP-slp$	0.91	0.43	0.9	1	$3.47\cdot10^{-15}$
$GP-tree$	1	1	1	1	1
$f_2(x,y,z,w)$	$SGPEA$	$TGPEA$	$BGPEA$	$GP-slp$	$GP-tree$
$SGPEA$	1	0.84	0.14	0.16	$8.16\cdot10^{-9}$
$TGPEA$	0.12	1	$2.53\cdot10^{-2}$	$3.07\cdot10^{-2}$	$4.29\cdot10^{-12}$
$BGPEA$	0.82	0.91	1	0.67	$3.77\cdot10^{-6}$
$GP-slp$	0.76	0.84	0.87	1	$1.38\cdot10^{-6}$
$GP-tree$	1	1	0.99	1	1
$f_3(x,y)$	$SGPEA$	$TGPEA$	$BGPEA$	$GP-slp$	$GP-tree$
$SGPEA$	1	0.41	0.96	0.28	$2.06\cdot10^{-2}$
$TGPEA$	0.32	1	0.84	0.37	$5.23\cdot10^{-3}$
$BGPEA$	0.35	$4.65\cdot10^{-2}$	1	$2.84\cdot10^{-2}$	$1.29\cdot10^{-2}$
$GP-slp$	0.44	0.51	0.99	1	$5.86\cdot10^{-2}$
$GP-tree$	0.93	0.42	0.88	0.34	1

Table 6: Minimal, median and maximal values of the validation fitness for each method and target function.

$f_1(x,y,z)$	min	med	max
$SGPEA$	0	1	94412.42
$TGPEA$	0	1	$4.74\cdot10^{7}$
$BGPEA$	0	1	$2.8\cdot10^{7}$
$GP-slp$	0	1	$3.75\cdot10^{7}$
$GP-tree$	$3.91\cdot10^{-24}$	$3.38\cdot10^{7}$	$1.65\cdot10^{8}$
$f_2(x,y,z,w)$	min	med	max
$SGPEA$	10.53	156.64	710.84
$TGPEA$	17.3	148.17	484.58
$BGPEA$	21.76	193.92	621.22
$GP-slp$	2.4	184.95	723.69
$GP-tree$	18.73	359.32	859.69
$f_3(x,y)$	min	med	max
$SGPEA$	0.36	0.49	0.71
$TGPEA$	0.41	0.49	0.62
$BGPEA$	0.41	0.48	0.66
$GP-slp$	0.41	0.48	0.72
$GP-tree$	0.39	0.51	1.01

Table 7: Values of means and variances.

$f_1(x,y,z)$	μ	σ
$SGPEA$	1345.95	10899.09
$TGPEA$	$6.11\cdot10^{6}$	$1.37\cdot10^{7}$
$BGPEA$	$1.98\cdot10^{6}$	$5.09\cdot10^{6}$
$GP-slp$	$6.5\cdot10^{6}$	$1.11\cdot10^{7}$
$GP-tree$	$4.72\cdot10^{7}$	$4\cdot10^{7}$
$f_2(x,y,z,w)$	μ	σ
$SGPEA$	211.05	150.71
$TGPEA$	175.2	116.59
$BGPEA$	221.29	133.87
$GP-slp$	229.7	157.18
$GP-tree$	362.58	160.8
$f_3(x,y)$	μ	σ
$SGPEA$	0.5	$6.41\cdot10^{-2}$
$TGPEA$	0.5	$4.79\cdot10^{-2}$
$BGPEA$	0.49	$5.86\cdot10^{-2}$
$GP-slp$	0.51	$6.65\cdot10^{-2}$
$GP-tree$	0.55	0.13

6 CONCLUSIONS

We have designed several cooperative coevolutionary strategies between a GP and an EA. The genetic program evolves straight line programs that represent functional expressions whereas the evolutionary algorithm optimizes the values of the constants used by those expressions. Experimentation has been performed on several groups of target functions and we have compared the performance between the studied strategies. In all cases the computational effort is fixed to a maximum number of evaluations. The quality of the selected model after the execution was measured considering a validation set of unseen points randomly generated, instead of the sample set used for the evolution process. A complete statistical study of the experimental results has been done. It has been shown that the coevolutionary architectures are promising strategies that in many cases are better than the traditional GP algorithms without coevolu-

tion. We have also confirmed that the straight line program structure clearly outperforms the traditional tree structure used in GP to codify the individuals.

In future work we wish to study the behavior of our coevolutionary model without assuming previous knowledge of the length of the slp structure. To this end new recombination operators must be designed. On the other hand we could include some local search procedure into the EA that adjusts the constants. Also the fitness function could be modified, including several penalty terms to perform model regularization such as complexity regularization using the length of the slp structure. Finally we might optimize the constants of our slp's by means of classical local optimization techniques such as gradient descent or linear scaling, as it was done for tree-based GP in (Keijzer, 2003) and (Topchy and Punch, 2001), and compare the results with those obtained by the computational model described in this paper.

ACKNOWLEDGEMENTS

This work is partially supported by spanish grants TIN2007-67466-C02-02, MTM2004-01167 and S2009/TIC-1650.

REFERENCES

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 (ICTAI)*, pages 517–524.

Axelrod, R. (1984). *The evolution of cooperation*. Basic Books, New York.

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

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

Casillas, J., Cordon, O., Herrera, F., and Merelo, J. (2006). Cooperative coevolution for learning fuzzy rule-based systems. In *Genetic and Evolutionary Computation Conference (GECCO 2006)*, pages 361–368.

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

Heintz, J., Roy, M. F., and Solerno, P. (1990). Sur la complexite du principe de tarski-seidenberg. In *Bulletin de la Societe Mathematique de France, 118*, pages 101–126.

Hillis, D. (1991). Co-evolving parasites improve simulated evolution as an optimization procedure. In *Artificial Life II, SFI Studies in the Sciences Complexity 10*, pages 313–324.

Keijzer, M. (2003). Improving symbolic regression with interval arithmetic and linear scaling. In *Procceedings of EuroGP 2003*, pages 71–83.

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

Maynard, J. (1982). *Evolution and the theory of games*. Cambridge University Press, Cambridge.

Michalewicz, Z., Logan, T., and Swaminathan, S. (1994). Evolutionary operators for continuous convex parameter spaces. In *Proceedings of the 3rd Annual Conference on Evolutionary Programming*, pages 84–97.

Rosin, C. and Belew, R. (1996). New methods for competetive coevolution. In *Evolutionary Computation 5 (1)*, pages 1–29.

Ryan, C. and Keijzer, M. (2003). An analysis of diversity of constants of genetic programming. In *Procceedings of EuroGP 2003*, pages 409–418.

Schwefel, H. P. (1981). *Numerical Optimization of Computer Models*. John Wiley and Sons, New-York.

Topchy, A. and Punch, W. F. (2001). Faster genetic programming based on local gradient search of numeric leaf values. In *Proceedings of the 2001 conference on Genetic and Evolutionary Computation (GECCO)*, pages 155–162.

Vanneschi, L., Mauri, G., Valsecchi, A., and Cagnoni, S. (2001). Heterogeneous cooperative coevolution: Strategies of integration between GP and GA. In *Proc. of the Fifth Conference on Artificial Evolution (AE 2001)*, pages 311–322.

Wiegand, R. P., Liles, W. C., and Jong, K. A. D. (2001). An empirical analysis of collaboration methods in cooperative coevolutionary algorithms. In *Proceedings of the 2001 conference on Genetic and Evolutionary Computation (GECCO)*, pages 1235–1242.