Department of
AUTOMATIC
CONTROL
and SYSTEMS
ENGINEERING
UNIVERSITY of SHEFFIELD

# CONTROLLING TREE SIZE GROWTH IN GENETIC PROGRAMMING

**K. Rodríguez-Vázquez and P.J. Fleming**

*Department of Automatic Control and Systems Engineering*
*University of Sheffield, Mappin Street S1 3JD, Sheffield, U.K.*
*katya@acse.shef.ac.uk, P.Fleming@sheffield.ac.uk*



**Abstract**

This paper presents an approach to solve the parsimony, or tree size growth, problem in Genetic Programming (GP). The approach is formulated as a multiobjective optimisation problem where parsimony is included as one of the objectives. This Multi-Objective Genetic Programming (MOGP) method is tested using the 6-Multiplexer benchmark problem. The MOGP is shown to consistently perform better than approaches which include parsimony pressure, a penalty to the program size, as part of a single objective function. The approach also results in a considerable reduction in computational processing time as the population evolves toward more parsimonious tree-structured representations.

*Index terms*-- Genetic Programming, Multiobjective Optimisation, Parsimony.

**Research Report #746**

# Controlling Tree Size Growth in Genetic Programming

Katya Rodríguez Vázquez and Peter J. Fleming[1]

*Abstract*--**This paper presents an approach to solve the parsimony, or tree size growth, problem in Genetic Programming (GP). The approach is formulated as a multiobjective optimisation problem where parsimony is included as one of the objectives. This Multi-Objective Genetic Programming (MOGP) method is tested using the 6-Multiplexer benchmark problem. The MOGP is shown to consistently perform better than approaches which include parsimony pressure, a penalty to the program size, as part of a single objective function. The approach also results in a considerable reduction in computational processing time as the population evolves toward more parsimonious tree-structured representations.**

*Index terms*-- **Genetic Programming, Multiobjective Optimisation, Parsimony.**

## I. Introduction

The adaptive search algorithm called Genetic Programming (GP) was designed by Koza [14]. GP is an evolution-based search model that is a subclass of Genetic Algorithms (GAs) which evolves populations of hierarchically structured computer programs according to their performance on a previously specified fitness criterion. The main difference between GP and its predecessor GA is the fact that GP genotypes, or individuals, are programs which are not fixed in length or size. The maximum depth height of the parse tree of the program is specified *a priori* to constrain the search space but all solutions up to and including this maximum are considered. When genetic operators operate over the population of computer programs, the new genotypes differ from their parents in structure (size, shape and contents).

Usually, GP assigns fitness values by evaluating each computer program over a number of different fitness cases. These fitness cases are typically only a small finite sample of the entire domain space. However, for Boolean functions with a few arguments, and this is the case in this paper, it is practical to use all possible combinations of values of arguments as the fitness cases. Thus, GP should be able to generate a 100% correct solution, that is, when the program returns the correct value for all given fitness cases.

Because the structure of the program is not known *a priori*, different executions of the GP approach may produce correct solution programs differing in size and content. All these solutions are syntactically valid programs but some of them may possess a more complex structure. Therefore, the principle of parsimony plays an important role. This principle is founded on the fact that "things should be explained and expressed in a simple way". In the context of GP, parsimony means that a correct program is expressed in its simplest form using the shortest parse tree structure.

However, parsimony in GP has not been fully addressed. The aim of this research is to introduce and describe a means of dealing with the evolution of parsimonious computer programs. The GP approach described here is based upon a multiobjective Pareto-optimal fitness function formulation.

## II. Multiobjective Optimisation

Often, optimisation problems are characterised by more than one objective and cannot be adequately solved by a single objective optimisation technique. Such optimisation problems involving multiple objectives are called *Multiobjective* (or *Multicriteria*) optimisation problems.
Although Vilfrido Pareto (1848-1923) [17], a French-Italian economist and sociologist, established an optimality concept in the field of economics based upon multiple objectives, only recently has this concept (called *Pareto-optimality)* been applied effectively in numerical optimisation. A definition of Pareto-optimality and other general concepts arising in multiobjective optimisation problems are given below.

### A. Definition 1. Multiobjective Optimisation

Multiobjective (also called multicriteria, multiperformance or vector) optimisation is defined as the problem of finding:

> a vector of decision variables which satisfies constraints and optimises a vector function whose elements represent the objective functions. These

[1] Both authors are with the Department of Automatic Control and Systems Engineering, University of Sheffield, Mappin Street, Sheffield S1 3JD, U.K.

functions are usually in conflict with each other. Hence, the term "*optimise*" means finding such a solution which would give values for all the objective functions acceptable to the designer.

Expressing this definition mathematically, it can be stated as:

Find the vector $\bar{x}^* = [x_1^*, x_2^*, \ldots, x_n^*]^T$ which satisfies the m inequality constraints,

$$g_i(\bar{x}) \geq 0 \qquad i = 1,2,\ldots,m \qquad (1)$$

the $p$ equality constraints

$$h_i(\bar{x}) = 0 \qquad i = 1,2,\ldots,p \qquad (2)$$

and optimises the vector function

$$\bar{f}(\bar{x}) = [f_1(\bar{x}), f_2(\bar{x}), \ldots, f_k(\bar{x})]^T \qquad (3)$$

where $\bar{x} = [x_1, x_2, \ldots, x_n]^T$ is the vector of decision variables.

Constraints (1) and (2) define the feasible region **X** and any point $\bar{x}$ in **X** defines a feasible solution. The $k$ components of the vector $\bar{f}(\bar{x})$ represent the non-commensurable criteria which must be considered. The constraints $g_i(\bar{x})$ and $h_i(\bar{x})$ represent the restrictions imposed on the decision variables. The vector $\bar{x}^*$ denotes the optimal solution set.

Keeney and Raiffa [13] have pointed out that optimisation, in the traditional mathematical sense, is impossible where multiple objectives are involved. Hence, the solution is a set of alternative solutions rather than a single optimal solution. This is known as the Pareto-optimal or non-dominant set.

### B. *Definition 2. Pareto-Optimality or Non-Dominance.*

The concept of Pareto-optimality (or non-dominance) constitutes by itself the origin of research in multiobjective optimisation. In a multiobjective minimisation problem, a vector $\bar{x}^* \in \mathbf{X}$ is Pareto-optimal (or non-dominated) if and only if there is no vector $\bar{x} \in \mathbf{X}$ with the characteristic

$$f_i(\bar{x}^*) \leq f_i(\bar{x}) \qquad \forall \quad i \in \{1,\ldots k\} \qquad (4)$$

and, there is at least one $i \in k$ such that

$$f_i(\bar{x}^*) < f_i(\bar{x}). \qquad (5)$$

The set of all Pareto-optimal decision vectors is called the Pareto-optimal or admissible solution set of the problem. No member of this set is dominated by any other point, that is, no other point has a better set of objective function measures.

## III. Evolution-Based Multiobjective Decision Making Methods

Thus, from the definition of Pareto-optimality it is seen that there is no unique solution to the multiobjective optimisation problem. Multiobjective optimisation problems involve two different tasks: the search for the non-dominated solutions set and the multiobjective decision making part [9]. A single objective problem itself can exhibit a complex search space due to a multimodal or non-linear problem solution space. In multiobjective problems, the actual objectives to be considered may be in conflict, thus adding an extra degree of complexity to the problem.

Traditionally, these two aspects of multiobjective optimisation are treated separately and most of the existing approaches do not work on large search spaces. In contrast, evolution-based methods afford a means of addressing both search and multiobjective decision making parts of the problem.

A number of researchers have used evolutionary algorithms to simultaneously optimise multiple objectives. Goldberg [7] pointed out that Rosenberg [20] was involved in the first attempt at multiobjective optimisation using evolutionary algorithms. He suggested a multiple properties function for the simulation of a population of single-celled organisms. Although he only considered a single property in his simulation, it was the beginning for further multiobjective evolutionary approaches.

Since Rosenberg's work, different multiobjective evolutionary optimisation approaches have been proposed. All of these methods retain the same representation and evolution mechanisms (reproduction, crossover and mutation) as conventional single-objective evolution-based methods. The main differences between MO approaches are associated with the objective functions and fitness assignment. A classification of MO evolution-based methods shown in Figure 1, which is based upon how methods evaluate and assign the fitness to each individual. Hwang and Masud [10] provide a survey of existing multicriteria decision making methods. In the context of evolution-based methods, Fonseca and Fleming [4] and Horn [9] give an extended review of multiobjective approaches. Here, a brief description and some examples of the classification given in Figure 1 are discussed.

In approaches where multiobjective decision making takes place before the search, the DM expresses preferences by means of an aggregating function which combines individual objectives in a single utility function translating the problem into a single objective problem before performing the optimisation. Examples of this class of multiobjective evolution-based techniques are the approaches proposed by Syswerda and Palmucci [23] and Jacob, Gorges-Schleuter and Blume [12]. These are based

upon a linear combination of the objective functions (weighted sum method) in order to determine a preferred solution. The weighted sum method is defined as

$$min \sum_{i=1}^{k} \omega_i f_i(\bar{x}) \quad (6)$$

where $\omega_i$ are the weighting coefficients representing the relative importance of the objectives and it is usually stated as

$$\sum_{i=1}^{k} \omega_i = 1 \quad (7)$$

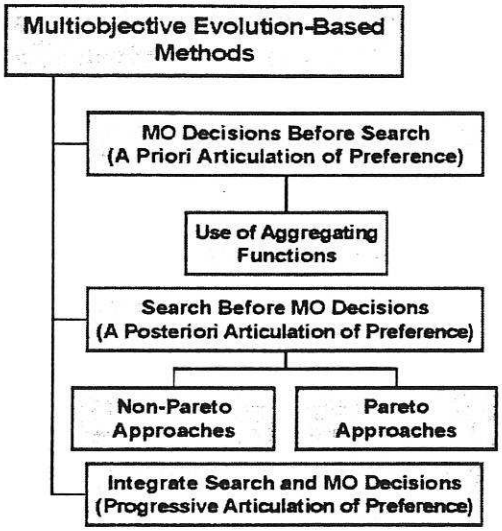


**Fig. 1.** Evolution-based multiobjective decision making taxonomy.

Horn and Nafpliotis [8] have proposed a different approach founded on a *MultiAttribute Utility Analysis (MAUA)*. The individual utility functions are here combined by multiplication rather than addition. The multiobjective approaches of Goldberg [7] and Richardson *et al.* [18] have their foundations in the concepts of handling constraints and penalty functions. Here, when a solution fails to meet a constraint, its utility function is assigned a large *penalty*. Nevertheless, quantifying or aggregating such a variety of objectives into a single criterion of choice, gives an undesirable reduction of reality.

In the case where the search is performed before the decision making, a set of candidate non-dominated solutions are identified. Non-Pareto (population-based) and Pareto approaches belong to this class of multiobjective methods. Here, the first attempt of treating objectives separately was proposed by Schaffer [21]. He introduced a method where appropriate fractions of the next generation, or sub-populations, were separately selected according to each objective. Genetic operations were performed after shuffling all the sub-populations together identifying non-dominated individuals. This approach is known as the Vector Evaluated Genetic Algorithm (VEGA). Fourman [5] suggested a multiobjective approach based upon a lexicographic ordering. The basic idea was to rank the objectives by assigning different priorities and lexically comparing individuals [2].

On the other hand, Pareto approaches are founded on a Pareto-ranking scheme. Goldberg [7] introduced a non-dominated sorting to rank the population according to Pareto-optimality. Non-dominated individuals are given rank one and then removed from the population. After this, the newly non-dominated individuals in the reduced population are assigned rank two and again, removed. This process continues until all individuals in the population are ranked. Using the concept of Pareto-optimality, Fonseca and Fleming [3] proposed a slightly different Pareto-ranking scheme. They ranked the population according to the *degree of domination*. Several Pareto-ranking approaches have been proposed but all of them have the basic principle of Pareto-optimality. A detailed description of the existing multiobjective Pareto-based evolutionary methods is given in Horn [9].

Finally, methods which integrate search and MO decision-making work by first searching to give the DM some knowledge about the possible trade-off range of the conflicting objectives. The DM then makes some multicriteria decisions to restrict and therefore, reduce the search space. The iterative process of searching/decision/searching continues until a single or small set of solutions is left.

Attempts at this iterative MO method have been carried out, but this area is still being explored. For example, Fonseca and Fleming [4] have extended the Pareto-based approach by combining it with a decision making stage. Their iterative multiobjective evolution-based technique is named the Multiobjective Genetic Algorithm (MOGA) which will be described in detail in the next section and extended into a Multiobjective Genetic Programming approach.

## IV. Multiobjective Genetic Algorithm

The Multiobjective Genetic Algorithm (MOGA) (Fonseca and Fleming [4]) covers three fundamental points in order to integrate a iterative search and multiobjective DM technique. These three aspects are described as follows.

### *1) Pareto-Ranking*

Firstly, MOGA uses a rank-based fitness assignment, where the rank of a certain candidate individual $x_i$ at generation $t$ corresponds to the number of individuals $p_i(t)$ in the current population by which it is dominated. This is expressed by

$$rank(x_i, t) = p_i(t) \quad (8)$$

All non-dominated individuals are assigned rank 0 and remaining individuals are penalised according to the above equation. Figure 2 expresses graphically this scheme by considering a bi-objective minimisation problem. Combining both objectives, the possible alternative solutions (Pareto frontier) are assigned a rank value of zero. It is seen that, for solutions with rank equal to zero, there is no other solution which is better in both objectives.

*2) Preference Information as Goal Vector*

In many practical optimisation problems, the solutions are constrained by a number of restrictions imposed on the decision variables. These constraints can often be seen and defined as hard objectives which need to be satisfied before the optimisation of the remaining "soft" objectives takes place.

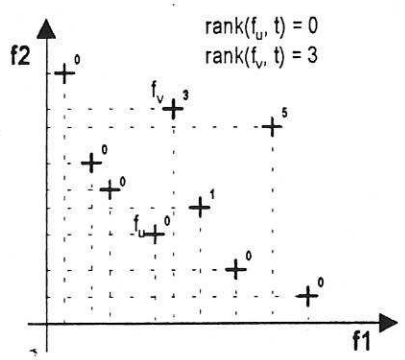


**Fig. 2.** Pareto-ranking scheme.

The introduction of goals enables us to address multiobjective problems which deal with constraints. Here, the constraints are incorporated into the vector function $\bar{f}$.

Introducing a desired level as preference information provides a means of evolving only a specific region of the search space. Figure 3 demonstrates this: the goal values $g_1$ and $g_2$ delimit the feasible solution area, reducing the search space and allowing the DM to focus on a region of the Pareto frontier. Hence, the rank-based fitness assignment is altered by preferring individuals inside the feasible space over those outside this region. This scheme allows discrimination between individuals (solutions) even though they are non-dominated but non-feasible solutions. In Figure 3, the same candidate solutions as shown in Figure 2 are considered but their rank positions are now modified under the assumption of preference (or goal) values. The light and dark areas determine the number of solutions that dominate $f_u$ and $f_v$, respectively.

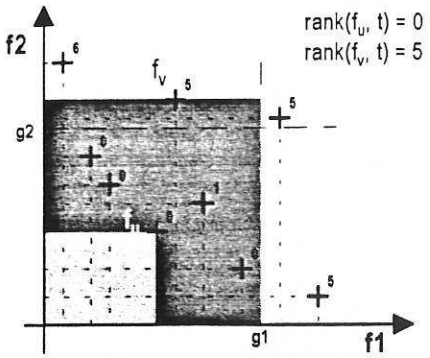


**Fig. 3.** Pareto-ranking with preference information: f1 and f2 have the same priority.

*3) Priority Information as Lexicographic Ordering*

Fonseca and Fleming [4] have also added to this approach a lexicographic ordering which has the aim of assigning different priority degrees. This means, the order in which the objectives will be optimised. Figure 4. shows, based upon the same bi-objective example, how the individuals are ranked when different priority levels are considered for each objective. Here, $\mathbf{f_2}$ (objective 2) possesses a higher priority than $\mathbf{f_1}$. Then, the individuals located inside the feasible region are ranked based on $\mathbf{f_1}$ because the goal value of $\mathbf{f_2}$, the objective with the higher priority, has completely been reached. In contrast, the rank assigned to non-feasible individuals is based on $\mathbf{f_2}$ and the number of feasible solutions. Thus, from Figure 4., $f_u$ has a ranking position equal to 2, whereas the rank assigned to $f_v$ has been increased to 7. Again, the light and dark areas show the number of solutions that dominate the points **u** and **v**.

Based on this bi-objective example, it can see that lexically ordering the objectives, a single optimal solution may emerge. However, when more than two objectives are considered, the optimisation becomes more complex and a single solution can no longer be obtained. The solution to the problem is then a reduced set of alternatives.

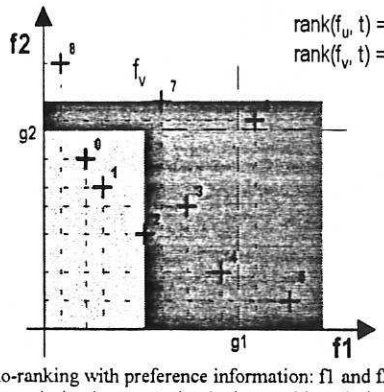


**Fig. 4.** Pareto-ranking with preference information: f1 and f2 exhibit different priority degrees (using lexicographic ordering).

Because the rank-based fitness assignment, including preference information and a lexicographic ordering scheme, take place in the objective function domain instead of the parameter domain, it can directly be applied to

genetic programming. Thus, the structure of MOGA can map into genetic programming by substituting the string-based and, generally, fixed-length representation with a hierarchical tree encoding.

## V. Multiobjective Genetic Programming

This section gives a review of initial attempts to develop multiobjective genetic programming and the alternative of combining genetic programming with the Pareto-ranking scheme. This multiobjective tool is also introduced as a means of controlling the growth tree problem.

### A. Correctness and Parsimony

GP is an evolutionary method designed to evolve and breed a population of individuals that consist of computer programs. Koza [14] states that when "getting computer programs to solve problems without being explicitly programmed, the size, the shape and the structural complexity of the solution should not be specified in advance". GP, which is a method of program induction[1], should be able to generate a 100% correct program.

The explanation of correct and parsimonious (defined in the introduction) program induction is the basis of the first attempts of multiobjective genetic programming approaches.

### B. Previous Approaches Involving Measures of Parsimony

In the previous Section a review of the existing multiobjective evolution-based techniques, mainly based on multiobjective genetic algorithms, has been described. However, the field of GP and multiobjective optimisation has hardly been explored. A few approaches have been developed to control the problem of tree growth of computer programs. Most of these approaches are formulated as an aggregating function with the aim of generating parsimonious computer programs.

Koza [14] considered parsimony as a factor in determining the fitness value of a tree-structured individual. He included the number of nodes (tree size) and the number of correct fitness cases returned by the individual in the fitness function, and aggregated them in a weighted sum approach resulting in a percentage of the total fitness value.

Iba *et al.* [11] proposed a method for controlling tree growth by means of a Minimum Description Length (MDL) principle. The MDL for a GP tree is defined as:-

$$mdl = (Tree\_Coding\_Length) + (Exception\_Coding\_Length), \quad (9)$$

where *Exception_Coding_Length* contains a measure of performance. Thus, the fitness function of each program individual considers the MDL component and the component related to the fitness function (number of correct fitness cases).

A similar approach based on the MDL principle has been proposed by Zhang and Mühlenbein [25]. They describe an adaptive method for fitness evaluation to dynamically guide the GP to grow and prune (restrict or cut) program trees. Their approach evolves parsimonious solutions without incurring premature convergence. This MDL principle-based approach dynamically grows and prunes the program size by balancing the ratio of training accuracy to solution complexity.

As is known, the evaluation of the fitness function consumes most of the computation process time in GP. This is the basis of the Siegel and Chaffe [22] approach. They use computation time to influence selection in the GP process through the introduction of an "aggregate computation time ceiling".

### C. Multiobjective Genetic Programming

Langdon [15] has used GP for evolving data structures using a Pareto-based approach, however this work has no bearing on the treatment of parsimony. In our approach, Multiobjective Genetic Programming (MOGP), we have the facility to control the search process by introducing goals which restrict the solution space and promote the search inside the feasible solution problem area. Multiobjective genetic programming combines the Pareto-ranking scheme with the concept of genetic programming. The MOGP approach: a) addresses multiobjective optimisation problems which are formulated as program induction, and b) controls the growth of tree-structured individuals, which is reflected in a reduction of the computation time by evaluating shorter structures and avoiding redundant nodes.

The MOGP approach differs from MOGA in that the representation of an individual is now expressed as a hierarchical tree. In order to demonstrate how MOGP can be applied to control the tree growth problem, the 6 Boolean-multiplexer problem is studied.

## VI. MOGP in Practice

For the purpose of experimental analysis, the 6-multiplexer problem has been selected. Motivations for it are based upon the fact that i) it is a well-understood problem in electronic circuit design, ii) the search space for this

[1] Program induction is seen as the generation of a computer program from a given set of input-output pairs or formal specification of the behaviour.

problem is finite and well understood due to its logical nature, and iii) it is a benchmark problem that has been used by several GP researchers (Koza [14]; O'Reilly [16]).

### A. 6-Boolean Multiplexer Problem.

**Definition:** The problem of learning the 6-multiplexer function is the task of decoding an address encoded in binary and returning the binary data value of a register at that address. The input consists of $k$ (equal to 2 in this case) address bits $A_i$ and $2^k$ ($2^2 = 4$) data bits $D_j$ (see Figure 5).

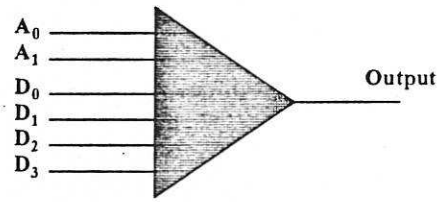


**Fig. 5.** The 6-multiplexer problem.

**Terminal Set:** The terminal set consists of the two address variables ($A_0$ and $A_1$) and the four variables ($D_0$, $D_1$, $D_2$ and $D_3$) representing the data registers. Then,

$\mathcal{T}$ = { $A_0$, $A_1$, $D_0$, $D_1$, $D_2$, $D_3$}

**Function Set:** The set of functions used in this problem are as follows.

- The conditional operator IF (condition, true-branch, false-branch) which takes three arguments. If the condition is true, it evaluates the true-branch, otherwise it goes to the false-branch.
- The logical OR (ARG1, ARG2) operator.
- The logical AND (ARG1, ARG2) operator.
- The unary NOT (ARG) operator.

### B. Multiobjective Fitness Evaluation.

The 6-multiplexer problem described here is a case study to demonstrate how MOGP may be used to produce correct and parsimonious Boolean programs. The problem is formulated as a multiobjective program induction problem where the objectives are defined as follows.

**Correctness:** There are $2^6 = 64$ possible outcomes (fitness cases) for the logic function represented in Fig.5, given the 6 input variables, as defined for the terminal set. The degree to which a MOGP individual or "program" correctly represents the correct behaviour is determined by counting the number of correct fitness cases arising from the 64 truth table combinations. Objective 1, $Obj_1$, is thus expressed as,

$$Obj_1 = 2^N - Number\_of\_Correct\_Fitness\_Cases$$

where N, the sum of address and data lines, is given by

$$N = k + 2^k .$$

$Obj_1$ is to be minimised. A program individual with an $Obj_1$ value equal to $2^k$ is the least fit. On the other hand, an individual with a value equal to zero is the most fit and is 100% correct.

**Input Variable:** This objective is introduced in order to give preference to such programs which involve *all* the address and data variables necessary to produce a correct program. In GP, many of the initial random individuals are not valid programs. An example is,
(IF (NOT $D_2$) $A_0$ $A_1$)
Here, GP does not make any distinction between an address or data variables. Another example is,
(IF (IF (IF $D_0$ $A_0$ $A_1$) $D_3$ $D_1$) $D_2$),
where a data variable is placed instead of an address variable and vice-versa. Nevertheless, this structure is preferred over the first example because it involves all the variables. The objective is concerned with the number of input variables used and we wish to maximise it. The syntactical validation of a program can be introduced as an objective in this multiobjective approach. However, it is not considered here.

**Parsimony:** This objective is defined as the number of nodes that comprises the parse tree.

These three objectives are summarised in Table I. Their priorities are also given. The MOGP tool allows three different priority levels which are *ignore*, *objective* and *constraint*. In fact, only one level is exercised in this example.

TABLE I. DESCRIPTION OF THE OBJECTIVES CONSIDERED FOR THE 6-MULTIPLEXER PROBLEM.

| | Objective | Description | Priority |
|---|---|---|---|
| 1 | Correctness | Number of correct answers (fitness cases) | Objective |
| 2 | Variables | Number of variables involved in the hierarchical solution individual | Objective |
| 3 | Parsimony | Size of hierarchical tree (number of nodes) | Objective |

### C. Results Analysis

This study demonstrates the capability of multiobjective genetic programming to simultaneously address correctness and parsimony aspects of the 6-multiplexer problem. This approach[2] is run for 6 different cases which differ in the population size and the mutation rate used. Population sizes of 100, 200 and 400 tree individuals and mutation rates of 0% and 10% were considered. In each

[2] This approach is implemented in MATLAB [23].

case, 50 runs of MOGP with different initial conditions were performed. The performance of single objective (SO) and multiobjective (MO) approaches are also compared, as shown in Tables II and III. In the single objective approach, the fitness function was defined to be simply the number of fitness cases that produce correct answers (see objective 1 from the *Multiobjective Fitness Evaluation* section above for more details).

First, note that the the single objective approach has not produced a single representation of the fully efficient parsimonious solution. Also, the frequency of generating a correct (non-parsimonious) solution is also lower for the single objective approach.

In contrast with the MO case, which addresses correctness and parsimony problems better with zero probability of mutation, the SO approach performs better when a non-zero mutation rate is used. In both cases, the success rate improves as the population size is increased, and the average number of generations needed to obtain a successful solution tends to be lower.

In the context of the MO approach and the average number of evaluations to obtain a successful solution, it is seen that when mutation is employed, the MO approach produces a lower number of average evaluations for a population size of 400 than 200.

In this example, the correct and fully efficient parsimonious solution consists of 10 nodes given by,

$$(\text{IF } A_0 \ (\text{IF } A_1 \ D_3 \ D_2) \ (\text{IF } A_1 \ D_1 \ D_0))$$

expressed hierarchically as,

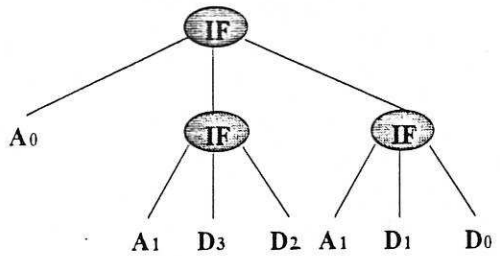


**Fig. 6.** Hierarchical encoding of the 6-multiplexer function.

Note that this is a simple benchmark problem where no preference information and priority levels have been considered. The multiobjective genetic programming approach has only been based upon the Pareto-ranking scheme.

### D. *Population Diversity*

In nature, diversification is a necessary condition for natural selection, and it is also important in artificial evolution.

For this 6 Boolean Multiplexer case study, two aspects of the problem have been analysed: correctness and parsimony. This problem has been formulated in a Pareto-optimal form and, therefore, the analysis of how population diversity behaves is defined in terms of both correctness and parsimony, as shown in Figures 7 and 8. These graphs are based on a run that considers a population size of 200 individuals, crossover and mutation probabilities of 90% and 10%, respectively. For this particular run, a 100% correct solution is reached at generation 87, and the fully efficient parsimonious solution at generation 108. However, the stopping criterion was set at 200 generations in order to observe how the population evolves.

As stated by Banzhaf *et al.* [1], genetic diversity is a necessary condition for rapid detection of a high-fitness individual and for efficient adaptation of the population to a changing environment. As seen from Figure 7, a set of individuals with a lower correctness fitness value appears before the population evolves to a 100% correct solution. After reaching the optimal solution, subsequent generations also maintain a certain degree of diversification.

With regard to the parsimony issue (Figure 8), it is seen that, at the first generation, complex tree representations are presented. However, more parsimonious trees emerge but complex ones remain at every subsequent generation.

Computational time is also measured and plotted as shown in Figure 9. Computational time expended for each generation rapidly falls to approximately one-third of the time spent evaluating the initial (randomly generated) population due to the pressure to seek out parsimonious solutions.

### E. *Performance Comparison with Other Approaches*

The 6-multiplexer problem is a well-known benchmark problem. Koza [14] used it in order to test approaches which include parsimony as a factor in fitness evaluation. He studied two different fitness functions which involve a parsimony factor. These have been formulated as aggregating functions differing only in the definition of the weights for performance (correctness) and parsimony.

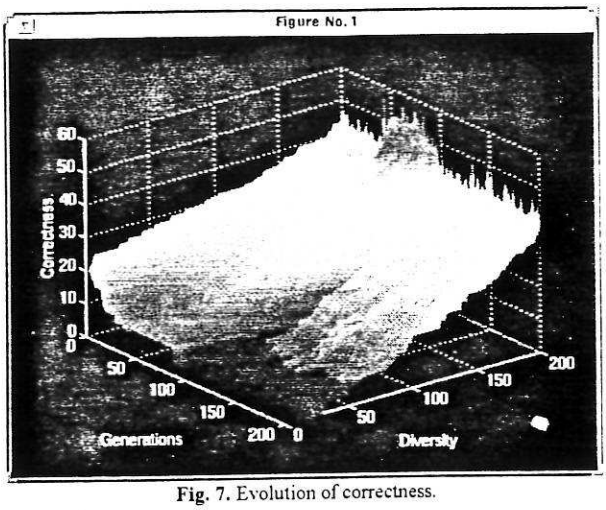


**Fig. 7.** Evolution of correctness.

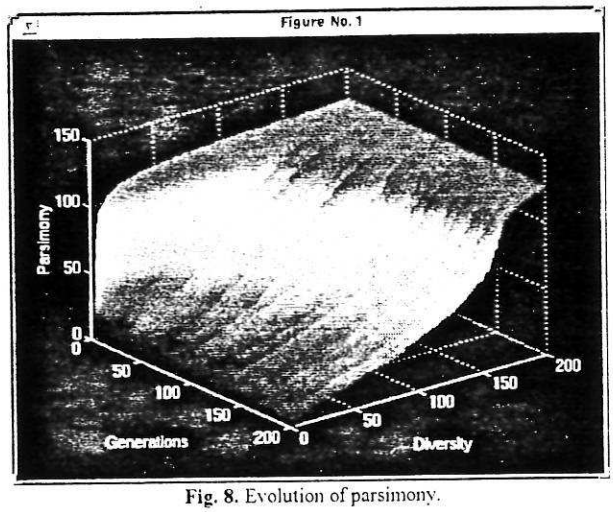


**Fig. 8.** Evolution of parsimony.

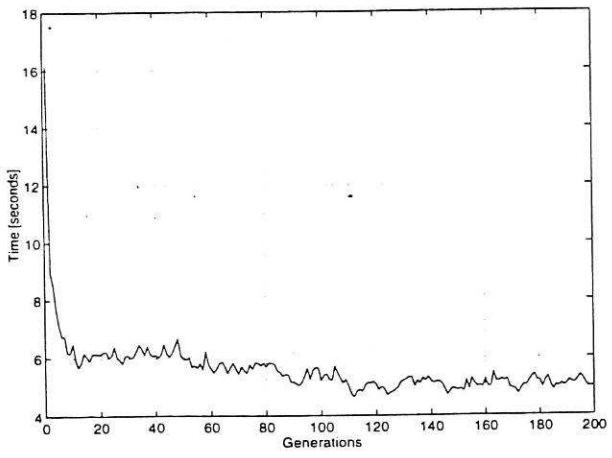


**Fig. 9.** Evaluation of computational processing time.

In a first approach, Koza defined the fitness to be the number of fitness casesfor which a tree individual gives the incorrect Boolean value plus the number of nodes in the hierarchical tree. From a total of 60 runs with a population of 1000 tree expressions using this approach, no fully efficient parsimonious expression emerged. The most parsimonious solution was an expression with 13 nodes and the probability of success was 38% for 50 generations.

In a second approach, the fitness was defined as the number of incorrect fitness cases when the best of the present generation scored fewer than 58. The fitness was redefined to be that of the previous approach when the best of the generation scored 58 or more correct fitness cases. Based on 30 runs, a correct expression containing 10 nodes emerged. However, the probability of success was even smaller, 27%.

The results provided using the MOGP approach (Tables II and III) has shown that, for the case of a population size of 400, there was a 95% success rate and, even for smaller population sizes, the success rate is higher than that obtained by either of Koza's approaches. There is also a higher probability of generating fully efficient parsimonious expressions, although the MOGP approach has been run for a larger number of generations. Nevertheless, Gathercole and Ross [6] have demonstrated that GP can perform even better with small populations over many generations than a small number of generations of large populations.

O'Reilly [16] has also worked on the 6-Multiplexer problem but she has focused her study on the effects of different genetic programming operators and hybrid genetic approaches. However, the success rate is still higher for the MOGP technique.

## VII. Conclusions

A Multiobjective Genetic Programming (MOGP) framework has been described, with particular attention paid to the generation of parsimonious solutions. To demonstrate its capability, MOGP has been tested on a simple problem and compared with existing genetic programming approaches which deal with parsimony within a single-objective (SO )approach.

The Boolean 6-multiplexer problem was the focus of the test study and MOGP was found to out-perform SO approaches. When compared with SO approaches which introduce parsimony pressure as a weighting fitness function, MOGP consistently produced correct and fully efficient parsimonious solutions. The SO approaches failed to produce a single representation of the fully efficient parsimonious solution.

While the MO approach treated the parsimony objective separately from other objectives, it was configured such that all objectives were addressed simultaneously in the evolution of improved solutions. Its treatment of parsimony also led to reduced computational effort as parsimony

pressure encouraged the generation of parsimonious (and, hence, computationally efficient) trees.

In the example considered here, no clear view emerged on appropriate settings for parameters such as crossover and mutation probabilities, population size and number of generations. For example, zero probability of mutation produced the best MOGP performance for the Boolean 6-multiplexer problem studied here. Future work will seek to extend this research, and that of [19], to generalise this means of controlling tree growth using MOGP, by testing it on a set of different benchmark problems.


## ACKNOWLEDGEMENTS

Katya Rodríguez Vázquez gratefully acknowledges the financial support of Consejo Nacional de Ciencia y Tecnología (CONACyT).


## REFERENCE


[1] Banzhaf, W., P. Nordin, R.E. Keller and F.D. Francone (1998) *Genetic Programming: An Introduction*. Morgan Kaufmann Publishers.

[2] Ben-Tal, A. (1980) Characterization of Pareto and Lexicographic Optimal Solutions. In Multiple Criteria Decision Making. Theory and Applications, (Fandel and Gal, Eds.) Springer-Verlag, pp. 1-11.

[3] Fonseca, C.M. and P.J. Fleming. (1993) An Overview of Evolutionary Algorithms in Multiobjective Optimization, Evolutionary Computation, 3(1), pp. 1-16.

[4] Fonseca, C.M. and P.J. Fleming (1998) Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms Part 1: A Unified Formulation. IEEE Trans Systems, Man and Cybernetics, 28 (1), pp 26-37.

[5] Fourman, M.P. (1985) Comparison of Symbolic Layout Using Genetic Algorithms. In Proceedings of the First International Conference on Genetic Algorithms (Grefenstette, Editor), pp. 141-153.

[6] Gathercole, C. and P. Ross (1997) Small Populations Over Many Generations Can Beat Large Populations Over Few Generations in Genetic Programming. In *Proc. of the Second Annual Conference on Genetic Programming* (Koza *et al.*, Editors), pp. 111-118.

[7] Goldberg, D.E. (1989) *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley.

[8] Horn, J. and N. Nafpliotis. (1993) *Multiobjective Optimization Using the Niched Pareto Genetic Algorithm*. IlliGAL Report 93005. Univeristy of Illinois at Urbana, Champaign.

[9] Horn, J. (1997) Multicriterion Decision Making. In *Handbook in Evolutionary Computation* (Bäck *et al.*, Editors), pp. F1.9:1-F1.9:15.

[10] Hwang, C.L. and A.S.M. Masud (1979) Multiple Objective Decision Making. Methods and Applications. Vol. **164** of *Lecture Notes in Economics and Mathematical Systems*. Springer-Verlag.

[11] Iba, H., H. de Garis and T. Sato. (1994) Genetic Programming Using a Minimum Description Length. In *Advances in Genetic Programming* (Kinnear, Editor) MIT Press, pp. 265-284.

[12] Jakob, W., M. Gorges-Schleuter and C. Blome. (1992) Application of Genetic Algorithms to Task Planning and Learning. In *Parallel Problem Solving from Nature, 2*. (Männer and Manderick, Editors), pp. 291-300.

[13] Keeney, R.L. and H. Raiffa (1976) Decisions with Multiple Objectives: Preferences and Value Trade-Offs. New York.

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

[15] Langdon, W.B. (1995) Evolving Data Structures with Genetic Programming. In *Proceedings of the Sixth International Conference on Genetic Algorithms* (Eshelman, Editor), pp. 295-302.

[16] O'Reilly, U-M (1995) *An Analysis of Genetic Programming*. PhD Dissertation. Ottawa-Carleton Institute of Computer Science, School of Computer Science, Carleton University.

[17] Pareto, V. (1896) *Cours D'Economie Politique*, Vol. **I** and **II**. F. Rouge, Lausanne.

[18] Richardson, J.T., M.R. Palmer, G. Liepins and M. Hilliard. (1989) Some Guidelines for Genetic Algorithms with Penalty Functions. In *Proceedings of the Third International Conference on Genetic Algorithms* (Schaffer, Editor), pp. 191-197.

[19] Rodríguez-Vázquez, K., C.M. Fonseca and P.J. Fleming (1997b) Multiobjective Genetic Programming: A Non-Lineat System Identification Application. *Late Breaking Paper at the Genetic Programming 97 Conference*. pg. 207-212.

[20] Rosenberg, R.S. (1967) *Simulation of Genetic Populations with Biochemical Properties*. PhD Thesis, University of Michigan.

[21] Schaffer, J.D. (1985) Multiobjective Optimization with Vector Evaluated Genetic Algorithm. In *Proceedings of the First International Conference on Genetic Algorithms* (Grefenstette, Editor), pp. 93-100.

[22] Siegel, E.V. and A.D. Chaffee (1996) Genetically Optimizing the Speed of Programs Evolved to Play Tetris. In *Advances in Genetic Programming*, Vol. 2. (Angeline and Kinnear, Editors), MIT Press, pp. 279-298.

[23] Syswerda, G. and J. Palmucci (1991) The Application of Genetic Algorithms to Resource Scheduling. In *Proc. of the Fourth International Conference on Genetic Algorithms* (Belew and Booker, Editors), pp. 502-508.

[24] The MathWorks (1992) MATLAB Reference Guide. The MathWorks, Inc.

[25] Zhang, B.T. and H. Mühlenbein (1996) Adaptive Fitness Functions for Dynamic Growing/Pruning. In *Advances in Genetic Programming*, Vol. 2. (Angeline and Kinnear, Editors), MIT Press.

TABLE II. EFFECTS OF POPULATION SIZE WITH A 0% MUTATION RATE, 90% CROSSOVER PROBABILITY, AND 500 GENERATIONS FOR THE 6-MULTIPLEXER PROBLEM (AVERAGED OVER 50 RUNS).

| | Single Objective | | | Multiobjective | | |
|---|---|---|---|---|---|---|
| **Population Size** | **100** | **200** | **400** | **100** | **200** | **400** |
| **Successes (Maximum = 50)** | 22 | 26 | 27 | 39 | 43 | 49 |
| **No. of Generations required to produce success** | 159.82 | 131.31 | 155.23 | 173.69 | 142.65 | 95.96 |
| **Average No. of Evaluations** | 15 982 | 26 262 | 62 092 | 17 369 | 28 530 | 38 384 |
| **Parsimony** | 0 | 0 | 0 | 11 | 12 | 22 |

TABLE III. EFFECTS OF POPULATION SIZE WITH A 10% MUTATION RATE, 90% CROSSOVER PROBABILITY, AND 500 GENERATIONS FOR THE 6-MULTIPLEXER PROBLEM (AVERAGED OVER 50 RUNS).

| | Single Objective | | | Multiobjective | | |
|---|---|---|---|---|---|---|
| **Population Size** | **100** | **200** | **400** | **100** | **200** | **400** |
| **Successes (Maximum = 50)** | 25 | 36 | 37 | 34 | 48 | 48 |
| **No. of Generations required to produce success** | 135.00 | 131.22 | 75.11 | 202.47 | 191.96 | 84.625 |
| **Average No. of Evaluations** | 13 500 | 26 244 | 30 044 | 20 247 | 38 392 | 33 850 |
| **Parsimony** | 0 | 0 | 0 | 5 | 12 | 17 |