Soft Comput (2007) 11:943–955
DOI 10.1007/s00500-006-0144-9

ORIGINAL PAPER

Crossover and mutation operators for grammar-guided genetic programming

Jorge Couchet · Daniel Manrique · Juan Ríos · Alfonso Rodríguez-Patón

Published online: 8 December 2006

Abstract This paper proposes a new grammar-guided genetic programming (GGGP) system by introducing two original genetic operators: crossover and mutation, which most influence the evolution process. The first, the so-called grammar-based crossover operator, strikes a good balance between search space exploration and exploitation capabilities and, therefore, enhances GGGP system performance. And the second is a grammar-based mutation operator, based on the crossover, which has been designed to generate individuals that match the syntactical constraints of the context-free grammar that defines the programs to be handled. The use of these operators together in the same GGGP system assures a higher convergence speed and less likelihood of getting trapped in local optima than other related approaches. These features are shown throughout the comparison of the results achieved by the proposed system with other important crossover and mutation methods in two experiments: a laboratory problem and the real-world task of breast cancer prognosis.

Keywords Grammar-guided genetic programming · Crossover · Mutation · Breast cancer prognosis

1 Introduction

Genetic programming (GP) is a means of automatically generating computer programs by employing operations inspired by biological evolution (Koza 1992). First, the initial population is randomly generated, and then genetic operators, such as selection, crossover, mutation and replacement are executed to breed a population of trial solutions that improves over time (Langdon and Poli 2001). The crossover operator bears most responsibility for the acceptable evolution of the genetic programming algorithm, because it governs most of the search process (evolution) (Escridge and Hougen 2004). The mutation operator produces small random changes in an individual to engender a new one and continue the search process. This prevents the loss of genetic diversity in the population, which is highly significant in the genetic convergence process (Lee and Yao 2004).

Grammar-guided genetic programming (GGGP) is an extension of traditional GP systems whose goals are (Whigham 1996; Wong and Leung 1995) to provide knowledge about the problem to be solved to simplify the search space and solve the closure problem. This problem involves always generating valid individuals (points or possible solutions that belong to the search space). This directly concerns the crossover and mutation operators, which are the ones that generate new individuals. To solve the closure problem, GGGP employs a context-free grammar (CFG), which establishes a formal definition of the syntactical restrictions of the problem to be solved and its possible solutions. Each of the individuals handled by GGGP is a derivation tree that generates and represents a sentence (solution) belonging to the language defined by the CFG (O'Neil and Ryan 2003; Manrique et al. 2005).

This paper proposes two new genetic operators for the GGGP paradigm. First, a crossover operator, called grammar-based crossover (GBC), boosts the exploration capacity of the genetic system when *ambiguous* grammars are used. This it does, because, unlike other

J. Couchet · D. Manrique (✉) · J. Ríos · A. Rodríguez-Patón
Facultad de Informática, Universidad Politécnica de Madrid, Campus de Montegancedo s/n. 28660 Boadilla del Monte, Madrid, Spain
e-mail: dmanrique@fi.upm.es

crossover operators, any node of the second parent able to generate valid offspring is eligible. The second operator, a mutation operator, called grammar-based mutation (GBM), is able to generate new valid individuals by randomly substituting an individual subtree for another randomly generated subtree that may have a different root.

The Sect. 7 shows that the use of these two operations jointly in a GGGP system provides faster convergence speed and is less likely to get trapped in local optima as compared with other commonly used crossover and mutation operators. Two experiments have been carried out. The first is the genetic programming of a laboratory problem. This test, which is not extremely complex, shows up the above-mentioned features of the proposed algorithms as compared with the others. The second experiment shows the successful application of the proposed GGGP system to the real-world problem of breast cancer prognosis. It involves analyzing suspicious masses and microcalcifications in breast tissue suspected of being carcinomas. The training and test patterns have been extracted from a database of real patients stored at a university hospital in Madrid.

2 Other related crossover and mutation operators

One of the first important GP crossover operators was defined by Koza (KX) (1992). This approach randomly swaps subtrees in both parent trees to generate the offspring. The main disadvantage of this operator is that individuals grow in size and complexity during the evolution process, which implies a very high computational cost, detracting enormously from convergence speed (Terrio and Heywood 2002). This effect is known as bloat or code bloat and is produced by an excessive exploration capability of the crossover (Manrique et al. 2006).

The strong context preservative crossover operator (SCPC) was proposed to preserve the context in which the subtrees appear in the parent trees and control the code bloat (D'haesler 1994). To achieve this goal, we have to define a system of coordinates for each of the nodes of the parent trees. Only those nodes that have the same coordinates are possible candidates to be chosen for crossing (subtrees swapping). It may occur with this approach that a parent subtree never changes position, which makes it impossible to move certain building blocks to other parts of the tree. This encourages building blocks to evolve in a specific region of the search space independently, increasing the *exploitation* capability of the search process a lot and thus multiplying the likelihood of getting trapped in local optima (Barrios et al. 2003).

There are also studies about the origin of the bloat phenomenon. Some research works state that there are more large than small individuals with good fitness in the search space. Therefore, large and well-adapted individuals are highly likely to be selected for crossing (Langdon and Poli 1997). Following this line of reasoning, it has been observed that the fitness of the individuals is penalized when large subtrees are eliminated; however, this does not occur when the subtrees introduced (called *introns*) do not improve the solution to the problem to be solved (Soule and Foster 1998). The notion of intron stems from microbiology and has been translated to evolutionary computation, where introns are segments of code within an individual that neither add nor detract from its fitness. However, introns are on the side of the convergence process as they improve the likelihood of preserving well-adapted building blocks. Recent studies report that the bloat phenomenon originates from the use of an inadequate crossover operator that chooses deeper and deeper crossover nodes as the algorithm evolves. This effect produces an offspring that is larger than its parents (Luke 2000b). Current research concerns the design of new algorithms capable of regulating tree growth (Panait and Luke 2004; Silva and Almeida 2003).

One of the most representative crossover operators, specially designed to prevent code bloat, is the *Fair* crossover (Crawford-Marks and Spector 2002), which is a modified version of the operator proposed by Langdon (1999). The Fair crossover works as follows. First, a crossover node in the first parent is selected randomly, and the length l of the subtree from this crossover node to the leaves is calculated. Then, a node from the second parent is selected randomly, and the length of the second subtree l_2, is calculated. If l_2 is in the range $[l - l/4, l + l/4]$, then the two subtrees are swapped. Otherwise, a crossover node in the second parent is selected randomly and the same check is run again. After n unsuccessful attempts, the subtree whose length is closer to the range $[l - l/4, l + l/4]$ is selected as the crossover node of the second parent. The advantage of this operator over Langdon's crossover is that it yields similar results in terms of code bloat limitation and is simpler to implement. But both have two key disadvantages: (a) the size distribution of the subtrees replaced in the population is not uniform, and (b) its exploration capability is low because of an excessive control of tree size. This slows down the system in its progress towards larger areas of the search space and, therefore, increases the time it takes to find solutions to which it can converge.

One of the most representative crossover operators for working with GGGP was proposed by Whigham (WX) (1995). It is still now in use because of its good performance (Grosman and Lewin 2004; Hussain 2003; Rodrigues and Pozo 2002). This operator, which assures the generation of valid offspring, randomly chooses a node representing a non-terminal symbol, called crossover node, from the first parent (derivation tree). Then, another node labeled with the same non-terminal symbol is selected in the second parent. Finally, both subtrees below the selected nodes are swapped. However, this operator has the disadvantage of not properly exploring the search space when ambiguous CFG are used (Hoai and McKay 2002). In this case, there are more non-terminal symbols, apart from the one selected in the first parent, that can be selected in the second parent to perform the crossing and generate valid offspring.

GGGP usually employs the *Standard* mutation operator, which substitutes the subtree whose root is the mutation node for another subtree whose symbol in the root node coincides with the one in the mutation node (Wong and Leung 2000). This constraint on matching non-terminal symbols has a negative impact on the exploration capacity of the operator when an ambiguous CFG is used.

3 Theoretical background

A context-free grammar G is defined as a string-rewiring system comprising a 4-tuple $G = (\Sigma_N, \Sigma_T, S, P)/\Sigma_N \cap \Sigma_T = \emptyset$, where Σ_N is the alphabet of non-terminal symbols, Σ_T is the alphabet of terminal symbols, S represents the start symbol or axiom of the grammar, and P is the set of production rules, written in BNF (Backus-Naur Form).

Based on this grammar, the individuals that are part of the genetic population are defined as derivation trees, where the root is the axiom S, the internal nodes contain only non-terminal symbols and the external nodes (leaves) contain only terminal symbols. A derivation tree represents the series of derivation steps that generate a sentence, which is a possible solution to the problem. Therefore, an individual codifies a sentence of the language generated by the grammar as a derivation tree. A sentence is ambiguous if it can be obtained by different derivation trees. A grammar *G is ambiguous* if any sentence that belongs to the language defined by the grammar is ambiguous.

$$G = (\Sigma_N, \Sigma_T, S, P):$$
$$\Sigma_N = \{S, E, F, N\}$$
$$\Sigma_T = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, -, =\}$$

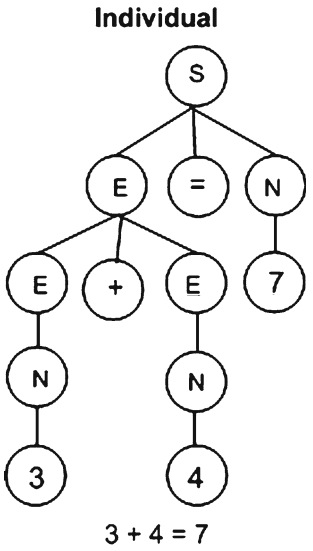

Fig. 1 An individual representing the sentence 3 + 4 = 7 of the grammar Eq. 1

$$\begin{aligned} P = \{&S ::= E = N; E \\ &::= E + E|E\text{-}E|F + E|F - E|N; F ::= N \\ N ::= &0|1|2|3|5|6|7|8|9\} \end{aligned} \quad (1)$$

As an example, which, for clarity's sake, will be used throughout this paper, a CFG is defined in Eq. 1. This CFG represents arithmetic equalities with sums and subtractions of one-digit integer numbers on the left side of the equality and a one-digit integer number on the right side.

Figure 1 shows the representation of the derivation tree of the individual 3 + 4 =7, which belongs to the grammar G defined in Eq. 1.

The grammar is ambiguous because there are sentences for which it is possible to find more than one derivation tree. Figure 2 shows that sentence 3 + 4 = 7 can be obtained from two different derivation trees.

Any GGGP system is able to find solutions to any problem whose syntactic restrictions can be formally defined by a context-free grammar. An appropriate evaluation function F, should be defined. This provides a fitness value for each of the possible solutions generated to drive the search process. The input for F is any of the individuals of the population, and its output provides a fitness measure that indicates how good the solution codified by the individual is for the problem at hand.

The evaluation function taken for the proposed example calculates the absolute value of the difference between the left and right sides of the expression so the optimal solution has a fitness of 0, which is, for example, the case of the individual 3 + 4 = 7.

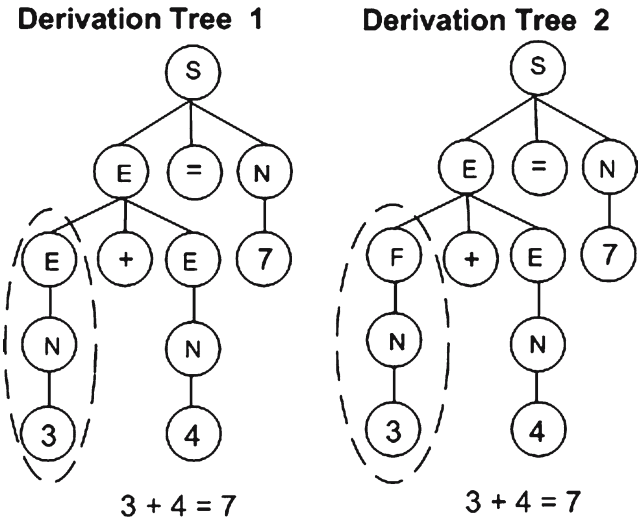

Fig. 2 Representation of two different derivations for the sentence 3 + 4 = 7

4 The proposed GGGP system

The structure of the new GGGP system is shown in Fig. 3 and consists of two modules: the evolution engine and the evaluation. The input for the *evolution engine module* is the grammar that defines the search space of the problem. First, PCT2 is used to generate the random initial population of trees, guaranteeing that generated trees will be around an expected and previously established depth (Luke 2000a). This approach, defined in the context of traditional genetic programming, is designed to the fast creation of programs as LISP-like program-trees of function nodes. This algorithm always generates valid programs (trees) starting from a function set F, divided into two disjoint sets: terminals (leaf nodes) Σ_T, and non-terminals (interior nodes) Σ_N. This algorithm has been easily adapted to the proposed GGGP system by randomly (same selection probability) executing the production rules defined in the CFG.

Once we have the initial population, the evolution engine module employs the genetic operators (selection, crossover, mutation and replacement) to search for the optimal solution, successively generating populations whose individuals are more and more adapted to the problem to be solved.

The selection operator employs the tournament method. Then, the proposed GBC, detailed in Sect. 5, is applied with a previously established probability to engender two new individuals (derivation trees) from two parents. GBM, detailed in Sect. 6, is applied to the offspring with a given probability. Finally, the SSGA (Grosman and Lewin 2004) replacement operator takes the individuals in that population that represent the worse solutions and replaces them with better-adapted offspring.

The input for the evaluation module is an evaluation function used to calculate the fitness of each of the individuals generated and the mean fitness of the entire population. To do so, a decodification process is implemented. This process involves concatenating the terminal symbols included in the leaves of the derivation tree to get the original sentence. The output of the evaluation module is the fitness of each individual.

5 Crossover operator: grammar-based crossover

The grammar-based crossover operator (GBC) is a general-purpose operator to solve problems with the GGGP paradigm. GBC has three significant characteristics: it prevents code bloat, provides an adequate trade-off between search space exploration and exploitation capabilities and, finally, is able to improve convergence speed by taking advantage of the main feature of ambiguous grammars: the existence of more than one

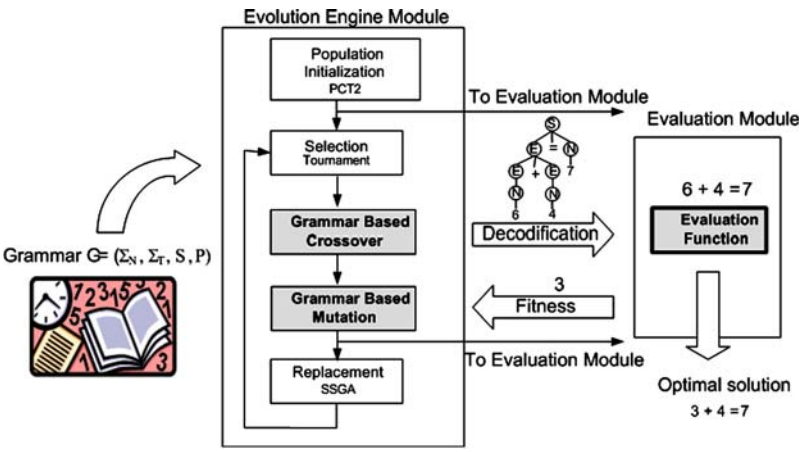

Fig. 3 The grammar-guided genetic programming system schema. The algorithms and operators proposed are represented as *grey boxes*

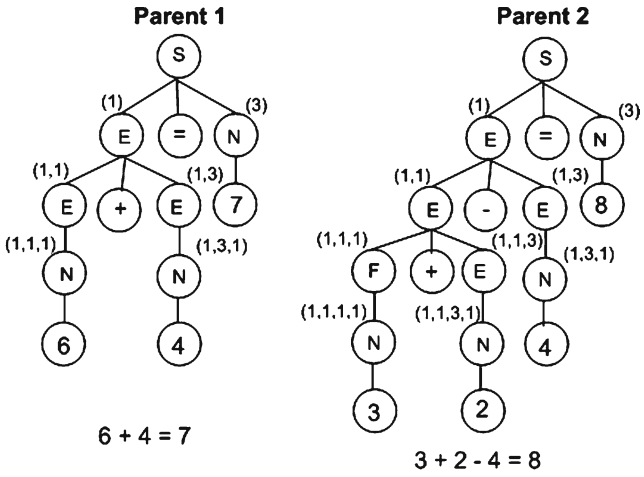

Fig. 4 Two individuals that belong to the grammar of Eq. 1. The nodes with non-terminal symbols are labeled with their respective coordinates

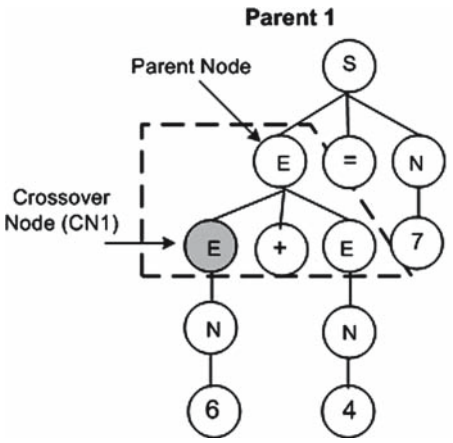

Fig. 5 Crossover node (*grey*), parent node (*arrow*) and main derivation (*dashed line*)

derivation tree for a single sentence. All these characteristics together provide GBC with a high convergence speed in terms of number of generations needed to reach the convergence and less likelihood of getting trapped in local optima, what leads to better solutions.

As a result, GBC produces two new valid individuals (derivation trees), starting from another two (parents) taken from the population by means of thirteen steps described below. An example is also given detailing the steps of how to apply the proposed crossover operator to two derivation trees, shown in Fig. 4. These trees correspond to two individuals that belong to the grammar of Eq. 1.

(1) Identify each of the nodes of the derivation trees that contain non-terminal symbols, except the root, by means of the tuple $N = (Z, \text{coord})$, where Z is the non-terminal symbol and coord is the coordinate of the node in the derivation tree. This is the notation used by the SCPC crossover operator. Label one of the two derivation trees as first parent and create the NT set (non-terminals) formed by the labeled nodes within this first parent. Figure 4 shows the derivation trees labeled with their node coordinates.

(2) If NT $\neq \emptyset$, then select an element of this set at random. This element is called *crossover node* (CN1). If NT $= \emptyset$, then it is not possible to cross the two parents in any other place except the root node. This results in two new offspring that are identical to the parents. For the proposed example, suppose that CN1 = $(E, (1, 1))$, shaded grey in Fig. 5, has been chosen at random.

(3) Get the parent node of the crossover node. This parent node has always a non-terminal symbol A, since we are dealing with a CFG. This symbol is the antecedent of one or more production rules of the grammar. The consequents of all productions whose antecedent is A are stored in the array R. As shown in Fig. 5, the parent node of $(E(1, 1))$ has the non-terminal symbol E and the consequents of all the production rules in which E is an antecedent in the Eq. 1 grammar are stored in R. Therefore, $R = [E + E, E - E, F + E, F - E, N]$.

(4) Calculate the tuple $T = (l, \text{p}, \alpha)$, where α is the consequent of the *main derivation* $(A ::= \alpha)$, referred to as the derivation that produces the parent node of CN1 in the first parent's derivation tree. The *length of the main derivation* l, is defined as the number of terminal and non-terminal symbols included in α. And p is the *position* of the crossover node in the main derivation. The tuple is $T = (3, 1\text{st}, E + E)$ in this case.

(5) Remove from the array R all those consequents whose length is different from *1*. Therefore, $R = [E + E, E - E, F + E, F - E]$ in the example.

(6) For each element of R, compare all its symbols with those of the consequent of the main derivation except for the one whose position (p) is the same as the crossover node. Then remove from R all those consequents in which any difference has been detected.
The consequents $E - E$ and $F - E$ are eliminated from R in the case in point, since the terminal symbol "-" is not present in the main derivation and hence $R = [E + E, F + E]$.

(7) Calculate the set X, formed by all those symbols of the consequents within R that are in position p. In this case, $X = [E, F]$.

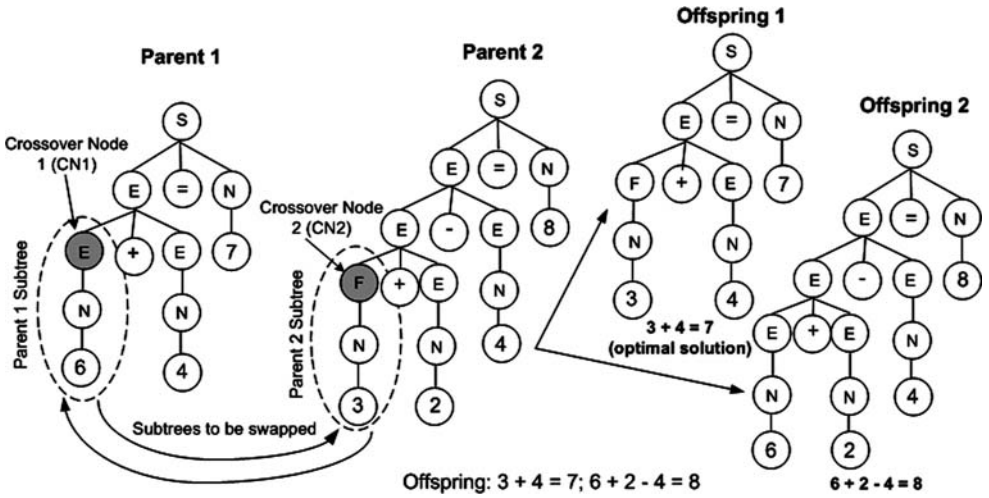

Fig. 6 An example of the grammar-based crossover operator

(8) If $X \neq \emptyset$, then select a symbol CS randomly. Then calculate the set PN formed by all the nodes of the second parent containing the symbol CS. If $X = \emptyset$, then there are no possible crossover nodes in the second parent that generate valid individuals with CN1. Therefore, remove this symbol from the NT set and go back to step 2.
In the proposed example, the crossover node of the second parent is generated by randomly choosing a node whose non-terminal symbol is E or F. Supposing that we choose CS $= F$, we calculate the PN set that is composed of all the nodes of the second parent containing the non-terminal F, PN = {(F,(1,1,1))}.

(9) If PN $\neq \emptyset$, choose a CN2 element randomly, which corresponds to the crossover node of the second parent. If PN $= \emptyset$, then remove CS from the set X and go back to step 8.
Following the example, there is only one possible choice of crossover node for the second parent: CN2 $= (F, (1, 1, 1))$. This node is shaded grey in Fig. 6.

(10) Calculate $P1$ as the sum of the depth of node CN1 in the first parent plus the depth of the subtree whose root is CN2 in the second parent. In this context, the depth of a node is measured as the number of levels existing from the root to this node, excluding the root. So, node CN1 is at depth 2. The other term used is the depth of a tree, measured as the number of levels there are in the tree from the root to the deepest leaf. So, the tree in Fig. 1 has depth 4.
Similarly as $P1$, calculate $P2$ as the sum of the depth of node CN2 in the second parent plus the depth of the subtree whose root is CN1 in the first parent. If $P1$ or $P2$ exceed the value of the maximum permitted depth for an individual (D), then remove CN2 from the PN set and go back to step 9.
In the case under study, $P1 = 2+2$ and $P2 = 3+2$ are calculated. Suppose that the permitted maximum depth is $D = 5$, then $P1, P2 \leq 5$. Therefore, it is possible to cross the nodes CN1 and CN2.

(11) If the non-terminal symbols of CN1 and CN2 match, then generate the two new offspring by swapping the two subtrees whose roots are CN1 and CN2.

(12) Otherwise, get the derivation generated by the parent node of CN2 in the tree and substitute the symbol CN2 for the symbol CN1 in this derivation.
Since the non-terminal symbol of CN1 is different from that of CN2, the derivation generated by the parent node of CN2, $E ::= F + E$, must be calculated. Then substitute the symbol F for the non-terminal symbol of CN1, thus giving the consequent: $E + E$.

(13) If the derivation resulting from the previous step matches any consequent of the grammar production rules, then the crossing can be done by swapping the two subtrees whose roots are CN1 and CN2. Otherwise, remove CN2 from the PN set and go back to step 9.

Since the consequent $E + E$ output in step 12 of our case study is a member of the set of consequents of the grammar production rules, all the requirements needed to cross nodes CN1 and CN2, which match the non-terminal symbols E and F, are met. Figure 6 shows the result of applying this operation.

The example given alongside this algorithm is a perfect illustration of GBCs ability to consider all possible nodes of the second parent that can generate valid individuals. This is a very significant characteristic, not shared by other important operators, such as WX, which prevents them from exploring points of the search space that could lead towards the sought-after solution. In the very example proposed, GBC is able to arrive at the final solution because it can choose the node with the non-terminal symbol F in the second parent. Conversely, none of the possible options of crossing with WX are optimal solutions of the problem, because it is compulsory to choose crossover nodes in the second parent with the non-terminal symbol $E: 3+2+4=7, 2+4=7, 3+2-4+4=7, 4+4=7; 6-4=8, 3+6-4=8, 3+2-6=8$ and $6=8$.

6 Mutation operator: grammar-based mutation

A new mutation operator, termed grammar-based mutation (GBM), is defined based on the algorithm proposed for the GBC. Given an individual (derivation tree) to be mutated, and having randomly chosen the node where the mutation is to take place (mutation node MN), the operator can substitute the subtree whose root is the mutation node for any other that yields a valid derivation tree as a result.

Like GBC, the proposed mutation operator has an important characteristic, which distinguishes it from other operators, and this is the possibility of changing the non-terminal symbol of the mutation node. This gives it a greater capacity of exploring the search space, thus helping to decrease the likelihood of getting trapped in local optima.

The input for the GBM consists of two parameters: the maximum permitted depth (D) to the individual once the mutation has taken place; and a list with the lengths of the grammar productions, which can be calculated when the initial population is generated. To do this, the following four definitions are given:

Definition 1 *The length of a terminal symbol $a \in \Sigma_T$ is 0. It is denoted $L(a) = 0$.*

Definition 2 *The length of a production rule that only derives terminal symbols is 1. It is denoted $L(A ::= a) = 1, \forall A \in \Sigma_N$ and $\forall a \in \Sigma_T^*$ (the closure of set Σ_T).*

Definition 3 *The length of a production rule $A ::= \alpha$ is the result of adding one to the maximum of the symbol lengths constituting the consequent. It is denoted $L(A ::= \alpha)$.*

Definition 4 *The length of a non-terminal symbol A is the minimum of the lengths of all its productions. It is denoted $L(A)$.*

The operator comprises fourteen steps. The first seven steps of the algorithm are similar to those described for GBC. The seventh step generates the set X, which is composed of non-terminal symbols that are the candidates for becoming the root of the mutated subtree. The algorithm is presented like GBC was, i.e. giving an example detailing the steps. This example illustrates the process of mutating the derivation tree of expression $6 + 4 = 7$ according to the Eq. 1 grammar and taking $D = 4$. Figure 7 sums up this process. It shows the list of production lengths of the grammar in Fig. 7a, whereas the mutation node is highlighted in grey in Fig. 7b. The first seven steps produce the same results as for the example proposed for GBC, that is, the set $X = [E, F]$. So, we continue with the eighth and subsequent steps:

(8) If X ≠ Ø, choose a symbol CS randomly. Otherwise, it is impossible to find a valid mutation in the node MN so, this node is removed from the NT set, and go back to step 2. Let CS = F, which is randomly chosen, for the proposed example.

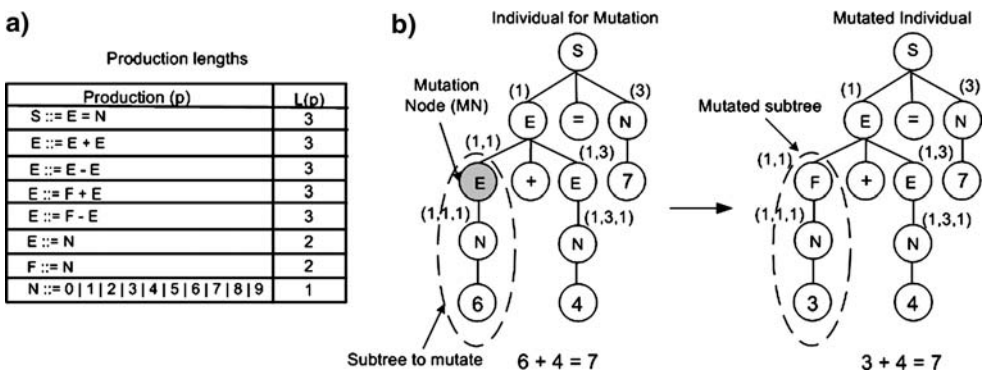

Fig. 7 Grammar-based mutation of the individual 6 + 4 = 7

(9) Calculate the mutation length ML, as the subtraction between the depth of the mutation node and the maximum permitted depth D. In our case, the depth of the mutation node is 2 and $D = 4$; therefore, $\mathrm{ML} = 4 - 2 = 2$.
(10) Assign value 0 to the current depth, $\mathrm{CD} = 0$.
(11) Get the set $\mathrm{PP} = \{\mathrm{CS}::=\alpha\}, \alpha \in \Sigma^* : \Sigma = \Sigma_N \mathrm{U} \Sigma_T$, such that $\mathrm{CD} + L(\mathrm{CS}::=\alpha) \leq \mathrm{ML}$. If $\mathrm{PP} = \emptyset$, then it is not possible to find a mutation to generate a valid individual with a depth less than D. Therefore, remove CS from the set X and go back to step 8.
For the case in point, $\mathrm{PP} = \mathrm{F}::=\mathrm{N}$, as it satisfies $\mathrm{CD} + L\ (F::=N) \leq \mathrm{ML}, 0 + 2 \leq 2$.
(12) Choose a production $\mathrm{CS}::=\alpha$ from PP at random. The only possibility in our example is to choose the production $F ::= N$ from the set PP.
(13) For each non-terminal $\mathrm{B} \in \alpha$, $\mathrm{B} \in \Sigma_\mathrm{N}$, calculate $\{\mathrm{B}::=\beta\}, \beta \in \Sigma^*$, such that $(\mathrm{CD} + 1) + \mathrm{L}(\mathrm{B}::=\beta) \leq \mathrm{ML}$. A derivation subtree, which does not exceed ML, is recursively obtained (step 13) from any production $\mathrm{B}::=\beta$.
Taking the only non-terminal symbol N from the production achieved in step 12, the set $\{N::=0, N::=1, N::=2, N::=3, N::=4, N::=5, N::=6, N::=7, N::=8, N::=9\}$ is generated because it satisfies $\mathrm{CD} + 1 + L(N ::= n) \leq \mathrm{ML}; 0 + 1 + 1 \leq 2$, for all $n = 0, 1, \ldots, 9$. Then the production $N::=3$ is chosen randomly. As this production derives only terminal symbols, the recursive process finishes and then we go on to step 14.
(14) The subtree whose root is the mutation node is substituted for the subtree calculated in steps 11–13.
In the example, the subtree whose root is the mutation node, represented by dashed lines in the derivation tree on the left of Fig. 7b, is substituted for the subtree generated in steps 11–13, illustrated with dashed lines in the derivation tree on the right of Fig. 7b. The new individual generated after the mutation process is 3 + 4 = 7, which is valid according to the grammar productions and is, also, a solution to the problem.

7 Results

In the following, we present and discuss the results achieved by the proposed operators, which are compared with other operators commonly used within the genetic programming paradigm.

To do so, two experiments were carried out: the search for arithmetical equalities (the example used throughout this paper), whose possible solutions are expressed by means of the grammar defined in Eq. 1, and a complex classification problem involving providing a breast cancer prognosis (benign or malignant) from the morphological characteristics of one of the most frequent type of lesions: masses. The same study has been conducted for microcalcifications, yielding similar results and findings.

For each experiment, a set of 100 independent runs was performed to give averaged results. In all cases, PCT2 was used to initialize the populations. After some tuning runs with all the different combinations of crossover and mutation operators, the best results were achieved with the following settings: the operator rates are 75% crossover and 5% mutation; tournament with a size of seven was the selection operator and SSGA was the replacement method. The same probability is assigned to each possible alternative when any of the crossover operators employed has several valid crossable symbols. This decision has been made to make the tests run comparable.

7.1 Search for arithmetical equalities

First, the results achieved by the algorithm in terms of convergence speed after applying the GBC crossover operator together with the GBM mutation operator were compared with the KX, Fair, SCPC and WX crossing operators, all of which were combined with the Standard mutation operator.

Table 1 shows the average number of generations needed to reach convergence and the standard deviation (SD) when using a population size of 50 and a maximum permitted depth of individuals (D) of 12.

Table 1 Generations needed for convergence

Crossover operator	Mutation operator	Averaged generations	Standard deviation
KX	Standard	26.14	6.85
Fair	Standard	16.72	3.69
SCPC	Standard	14.59	3.43
WX	Standard	13.26	3.45
GBC	GBM	11.60	3.41

Table 2 Post test score

Tukey HSD. Dependent variable: generations needed for convergence

Comparisons: (crossover, mutation)	Mean difference	Significance
(Fair, Standard), (GBC, GBM)	5.12	0.013
(SCPC, Standard), (GBC, GBM)	2.99	0.024
(WX, Standard), (GBC, GBM),	1.66	0.039

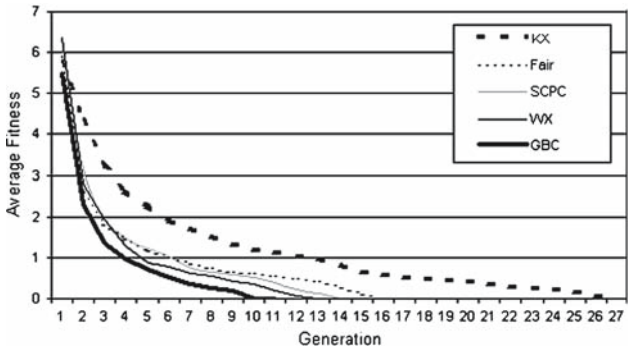

Fig. 8 Convergence speed for each crossover and mutation operator. Search for arithmetical equalities

Similarly, Fig. 8 shows the evolution process for the average fitness of the population under the same conditions.

Looking at the results, we find that KX is clearly the operator that provides the slowest convergence speed. This is due to its excessive exploration capacity resulting from performing the crossovers completely at random.

For the other four crossover operators, an analysis of variance has been run to show that the differences between means are actually meaningful. In this study, the combination of crossover and mutation operators serves as the independent variable, while the convergence speed, measured in number of generations needed to reach the convergence, is the dependent variable.

The null hypothesis (the means are equal) can be rejected because of the resulting $F(df = 3/396)$ of 30.235, whose significance level is $p < 0.01$, indicating that the average number of generations needed for convergence depends on the crossover and mutation operators employed.

As there are sizeable differences in convergence speed depending on the crossover and mutation operators employed, the Tukey HSD Test was used to make post hoc comparisons to demonstrate whether there are statistically significant differences between the proposed crossover and mutation operators (GBC and GBC) and the combinations of Fair, SCPC and WX crossovers with standard mutation. Table 2 shows the significance level for these multiple comparisons, taking into account that the significance level for the mean difference is $p < 0.05$.

From these statistical results (Tables 1, 2; Fig. 8) we find that SCPC has a high convergence speed in the early iterations due to its good exploitation capability. Later on, however, this characteristic slows down the convergence towards the optimum because of trapping in local optima. This slowdown increases when the SCPC works with small-sized populations. Quite the opposite applies to the Fair crossover operator: it performs better when working with small-sized populations. However, for populations of larger size, as shown in Fig. 8, it is slightly outperformed by SCPC. Whigham's operator (WX) together with the Standard mutation operator achieves quite good results. Its performance, however, is poorer than that delivered by the proposed operators. The combination of GBC and GBM offers the best results regarding convergence speed in all the tests performed. This is because GBC and GBM are able to take advantage of the grammar's ambiguity to explore all possible paths that lead to the sought-after solution.

GBC has also been combined with Standard mutation under the same conditions as described above. It yielded an average number of generations to convergence of 12.37 with SD = 3.39. This result shows that it is not Standard mutation that is hindering the performance of the other combinations, confirms that the crossover operator is most responsible for the evolution process and that GBM is on the side of GBC, improving the convergence speed.

Another significant aspect that can explain why the proposed operators work well is that the average depth of the individuals generated with KX, Fair, SCPC and WX is not evenly distributed. For $D = 12$, an average depth between 4.90 for KX and 5.66 for WX is achieved (discarding all the trees larger than 12). This means that the individuals generated tend to be small and are, therefore, not well distributed. On the contrary, GBC plus GBM gets an average tree size of 7.30 with a standard deviation of 3.18. This means that there is more diversity in size among the individuals that are being generated.

7.2 Breast cancer prognosis

The second experiment involved searching a knowledge base of fuzzy rules (Bojarczuk et al. 2004) that could give a prognosis of masses present in breast tissue suspected

Table 3 Misclassified cases in training and testing

Crossover operator	Mutation operator	Misclassified cases	
		Training	Testing
KX	Standard	105/265 (39.63%)	20/50 (40%)
Fair	Standard	101/265 (38.12%)	19/50 (38%)
SCPC	Standard	94/265 (36.48%)	18/50 (36%)
WX	Standard	73/265 (27.55%)	14/50 (28%)
GBC	GBM	58/265 (21.89%)	11/50 (22%)

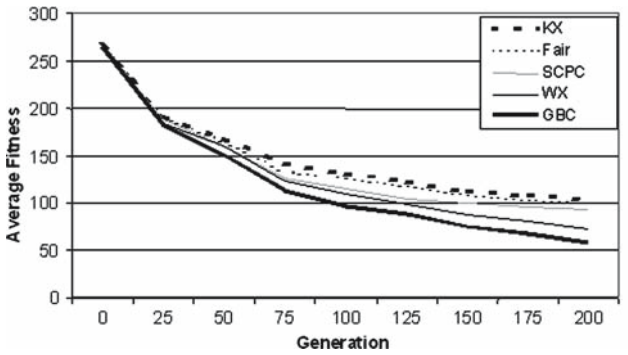

Fig. 9 Convergence speed for each crossover and mutation operator. Breast cancer prognosis

of being a carcinoma. This application is part of a larger project for automatically detecting breast pathologies. The input for the system built so far is a complete set of views of digitalized mammograms from both of the patients' breasts, which is used to search for suspicious masses, one of the main abnormalities that can be detected by mammography. The output is a set of characteristics for each abnormality found. These characteristics are stored in a database of 315 lesions of real patients aged from 25 to 79 years at a Madrid University Hospital. All these cases have been diagnosed by two expert radiologists, whose results were compared with the results of the biopsy, which is also available. We have used these clinical data to run the tests. From the 315 lesions available in the database, 265 were selected randomly to train the genetic programming system; the remaining 50 were used to perform tests with cases never before presented to the system. The characteristics for each mass stored are: the hospital expert radiologists' prognosis, the actual prognosis after carrying out a biopsy and the morphological characteristics that follow:

Size It represents the size of the mass, measured as the widest diameter in millimeters.

Morphology of the mass It has five possible values: architectural distortions, lobulated, oval, round and irregular.

Margins This characteristic describes the external limits of the mass. It has five possible values: circumscribed, ill-defined, microlobulated, spiculated and obscured.

Density It represents the texture of the tissue inside the mass. The possible values are: high, equal, low and fatty.

A fuzzy knowledge base to solve this problem required a CFG deliberately designed as ambiguous, containing 16 non-terminal symbols, 51 terminals and 55 production rules. The fitness value of an individual corresponds to the number of masses in which the prognosis inferred by the fuzzy knowledge base represented by this individual is different from the prognosis made after the biopsy. Table 3 illustrates the average number of cases incorrectly classified during the training and test phases, using a population size of 100 and $D = 20$.

Figure 9 shows the evolution process for the average fitness of the population and for each combination of crossover and mutation operators shown in Table 3.

WX gets better results than KX, Fair and SCPC crossover operators. This is because this operator guarantees the generation of valid offspring, which boosts convergence speed. This operator correctly classifies 72.45% (192/265) of the training lesions and 72% (36/50) of the test lesions. GBC plus GBM provides the best results and the swiftest convergence speed: 78.11% (207/265) of correct results in training lesions and 78% (39/50) in test lesions.

By way of an example, the following rule belongs to the fuzzy knowledge base found by the GP system using GBC plus GBM that best prognosticates suspicious masses: "IF margins = spiculated AND morphology = irregular, THEN prognosis = malignant". In this rule, *margins* and *morphology* are fuzzy variables that can take the qualitative values of spiculated and irregular, respectively. If so, then the system will give a prognosis of malignant.

Again, in this experiment, the depth of the trees that are being generated through the evolution process is better distributed using GBC and GBM than using the other approaches: an average tree size of 13.60 with a standard deviation of 5.34 is obtained for GBC and GBM, whereas the average tree size for the other operators ranges from 7.30 for KX to 8.12 for WX.

It is usual practice in the field of medicine to use three criteria to report statistical results and for the

Table 4 Accuracy, specificity and sensitivity for the masses prognosis problem for the set of test patterns given by two expert radiologists, the proposed GGGP system and the combination of KX, Fair, SCPC and WX with Standard mutation

Crossover	Mutation	Accuracy (%)	Specificity (%)	Sensitivity (%)
KX	Standard	60.32	61.29	59.37
Fair	Standard	61.90	64.05	59.88
SCPC	Standard	64.44	66.23	63.12
WX	Standard	71.43	73.86	70.44
GBC	GBM	78.09	80.39	75.93
Expert radiologists				
Doctor A		76.19%	66.30%	89.55%
Doctor B		69.84%	58.51%	86.61%

purpose of comparisons. These are accuracy, specificity and
sensitivity.

Accuracy is the percentage of correctly diagnosed cases compared to incorrect diagnoses:

$$\text{Accuracy} = (\text{T}_P + \text{T}_N)/\text{Number of instances}. \quad (2)$$

where T_P (True Positive) are the cases prognosticated as malignant that really are malignant and T_N (True Negative) are the cases prognosticated as benign that are benign.

Specificity represents the percentage of cases prognosticated as negative, benign lesions, over the total number of negative cases:

$$\text{Specificity} = \text{T}_N/(\text{T}_N + \text{F}_P). \quad (3)$$

where F_P (False Positive) is the number of cases classified as malignant when they really are benign.

Sensitivity is the percentage of cases prognosticated as positive, malignant lesions, over the total number of positive cases:

$$\text{Sensitivity} = \text{T}_P/(\text{T}_P + \text{F}_N). \quad (4)$$

where F_N (False Negative) is the number of cases classed as benign when they really are malignant.

Table 4 shows the results in terms of accuracy, specificity and sensitivity of the prognosis for the set of test patterns of masses, comparing the results of the proposed GGGP system with the Standard, Fair, SCPC, and WX crossover operators together with standard mutation and with the results of two expert radiologists specialized in image prognosis (called A and B).

The combination of GBC and GBM provides the best results in terms of accuracy when compared with the other crossover and mutation operators and it even offers a slight improvement of the most skilful radiologist's correctness rate. In addition, the proposed GGGP system outperforms the doctors as regards specificity though it is worse as regards sensitivity. This is because of the way a radiologist diagnoses a detected breast lesion: only when the doctor is really sure is the lesion classified as benign; if there is any doubt, the lesion is classified as malignant. The GGGP system has no such prejudices, which means that it provides better results than radiologists for true negatives, but also worse results for false negatives, which implies a bigger risk.

8 Discussion and conclusion

In this paper, we propose a GGGP system able to find solutions to any problem that can be syntactically expressed through context-free grammars. The proposed system includes two original operators specially designed for the GGGP paradigm: crossover (GBC) and mutation (GBM).

GBC and GBM have three main characteristics. First, they prevent code bloat as both of them incorporate mechanisms to limit the size of the offspring or mutated trees, respectively. Second, they boost (improve) the exploration capacity of the search space: GBC can choose among a higher number of nodes in the second parent, that is, any node able to generate valid offspring, whereas GMB can select non-terminal symbols for the root of the mutated subtree that differ from the non-terminal symbol of the mutation node. This feature is observed in the results section for the first of the two problems: an average tree size throughout the evolution process centered within the range of permitted values (which does not occur with the other operators) with a relatively high standard deviation indicates good genetic diversity. Finally, the third characteristic of the proposed operators is that they are able to benefit from the property of the ambiguous grammar consisting in having different derivation trees to represent the same sentence (solution). If an unambiguous grammar is used, results are similar to WX and Standard mutation operators. As was observed in the results section, the combination of these characteristics provides a high

convergence speed in number of generations and less likelihood of getting trapped in local optima as compared with other operators now in use. Although the tests run do not prove that GBC plus GBM performs better than the others in all cases, they do confirm that the presence of these three features in the proposed system improves the convergence process.

The proposed system has been applied to solve a straightforward problem: searching for arithmetical equalities. In this experiment, statistical significance has been reported to show that the proposed system takes fewer generations to converge than the other crossover and mutation combinations. Specifically, this convergence speed is improved by 12.5% with respect to WX and Standard operators, the second combination of operators that best results achieved. Besides, we have presented its application to a more complex problem: the prognosis of breast lesions: masses, which are one of the two most important signs to detect breast cancer through radiodiagnosis. Similar results have been achieved for microcalcifications. They represent the other important sign for detecting breast cancer, and have, therefore, not been reported. These tests have been carried out on lesions of real patients at the 12 de Octubre University Hospital in Madrid. The reported results show that the proposed system outperforms the other genetic operators, achieving an accuracy rate of 78.09%, a 6.66% improvement on WX plus Standard (71.43%). Accuracy is similar and specificity is better as compared to expert radiologists, although sensitivity is worse. This entails a greater risk for patients as the number of false negatives output by the proposed system is still high. This implies that the machine cannot (and is not intended to) substitute the human expert, but is suitable for use (and is being used) as a second opinion in a computer-aided prognosis system as part of screening processes.

The convergence speed tests have been measured in terms of number of generations to show that the proposed operators improve the convergence process. In return for fewer generations to convergence, GBC and GBM perform more operations (compared with the other operators) in each execution. In terms of time, then, the proposed system is on average 4% slower than WX with Standard mutation. However, this drop in performance is offset by a lesser likelihood of getting trapped in local optima that leads to an improvement in the solutions to problems, which is more evident for complex problems, as shown in the classification results for breast cancer prognosis.

The development of this work opens the way to further lines of research in the field of GGGP. One of them involves finding an algorithm that can estimate the maximum depth (D) of the trees generated throughout the evolution process assuring that the optimal solution is reached. This would overcome the disadvantage of not being able to find the solution to a problem because the maximum depth (D) chosen for the GBC and GBM operators is too restrictive.

At present, we are working on the definition and design of non-trivial problems to be solved by the different combinations of operators. The goal is to accomplish a statistical analysis to generalize the results. It would also be interesting to find out how changing the probabilities of genetic operator occurrence would affect the performance of the proposed and other systems.

Finally, we are also looking at how to define a measure of the ambiguity of a context-free grammar that can optimize the performance of the proposed system. An ambiguous context-free grammar provides more than one derivation tree that codifies the solution to the problem and increases the capacity of selection of the nodes to be crossed and mutated by the GBC and GBM operators. However, empirical tests demonstrate that if the grammar is too ambiguous, the search space increases considerably, and this causes a loss in the system performance. Therefore, our aim is to find a formal method that can establish the exact grammar ambiguity and achieve the maximum convergence speed possible.

Acknowledgments This research is being funded by the Spanish Ministry of Science and Education under project ref. DEP2005-00232-C03-03. We would also like to thank Fernando Marquez and Pablo Manrique for their valuable comments. Special thanks to the reviewers for their suggestions.

References

Barrios D, Carrascal A, Manrique D, Ríos J (2003) Optimization with real-coded genetic algorithms based on mathematical morphology. Int J Comput Math 80(3):275–293

Bojarczuk CC, Lopes HS, Freitas AA, Michalkiewicz EL (2004) A constrained-syntax genetic programming system for discovering classification rules: application to medical data sets. Artif Intell Med 30(1):27–48

Crawford-Marks R, Spector L (2002) Size control via size fair genetic operators in the pushGP genetic programming system. In: Proceedings of the genetic and evolutionary computation Conference, New York, pp 733–739

D'haesler P (1994) Context preserving crossover in genetic programming. In: IEEE Proceedings of the 1994 World Congress on computational intelligence, Orlando, 1:379–407

Escridge BE, Hougen DF (2004) Memetic crossover for genetic programming: evolution through imitation. Lect Notes Comput Sci 3103:459–470

Grosman B, Lewin DR (2004) Adaptive genetic programming for steady-state process modelling. Comput Chem Eng 28: 2779–2790

Hoai NX, McKay RI (2002) Is ambiguity useful or problematic for grammar guided genetic programming? A case of study. In:

Proceedings of the 4th Asia-Pacific conference on simulated evolution and learning, Singapore, pp 449–453
Hussain TS (2003) Attribute grammar encoding of the structure and behavior of artificial neural networks. PhD Thesis, Queen's University, Kingston, Ontario
Koza JR (1992) Genetic programming: on the programming of computers by means of natural selection. MIT, Cambridge
Langdon WB (1999) Size fair and homologous tree genetic programming crossovers. In: Proceedings genetic and evolutionary computation conference, GECCO-99, Washington DC, pp 1092–1097
Langdon WB, Poli R (1997) Fitness causes bloat. In: Proceedings of the 2nd online world conference on soft computing in engineering design and manufacturing, on-line, pp 13–22
Langdon W B, Poli R (2001) Foundations of genetic programming. Springer, London
Lee CY, Yao X (2004) Evolutionary programming using mutations based on the Lévy probability distribution. IEEE Trans Evol Comput 8(1):1–13
Luke S (2000a) Two fast tree-creation algorithms for genetic programming. IEEE Trans Evol Comput 4(3):274–283
Luke S (2000b) Code growth is not caused by introns. In: Proceedings genetic and evolutionary computation conference, GECCO 2000, Las Vegas, pp 228–235
Manrique D, Márquez F, Ríos J, Rodríguez-Patón A (2005) Grammar based crossover operator in genetic programming. Lect Notes Comput Sci 3562:252–261
Manrique D, Ríos J, Rodríguez-Patón A (2006) Evolutionary system for automatically constructing and adapting radial basis function networks. Int J Neurocomput, DOI 10.1016/j.neucom.2005.06.018 (in press)
O'Neil M, Ryan C (2003) Grammatical evolution: evolutionary automatic programming on arbitrary language (genetic programming). Chap. 2. Kluwer, Norwell
Panait L, Luke S (2004) Alternative bloat control methods. Lect Notes Comput Sci 3103:630–641
Rodrigues E, Pozo A (2002) Grammar-guided genetic programming and automatically defined functions. In Proceedings of the 16th Brazilian symposium on artificial intelligence, Brazil, pp 324–333
Silva S, Almeida J (2003) Dynamic maximum tree depth. A simple technique for avoiding bloat in tree-based GP. Lect Notes Comput Sci 2724:1776–1787
Soule T, Foster JA (1998) Removal bias: a new cause of code growth in tree based evolutionary programming. In: IEEE Proceedings of the International conference on evolutionary computation, Indianapolis, pp 181–186
Terrio MD, Heywood MI (2002) Directing crossover for reduction of bloat in GP. In: IEEE Proceedings of Canadian conference on electrical and computer engineering. 2:1111–1115
Whigham PA (1995) Grammatically-based genetic programming. In: Proceedings of the workshop on genetic programming: from theory to real-world applications, California, pp 33–41
Whigham PA (1996) Grammatical bias for evolutionary learning. PhD Thesis, School of Computer Science, Australian Defence Force (ADFA), University College, University of New South Wales
Wong ML, Leung KS (1995) Applying logic grammars to induce sub-functions in genetic programming. Evol Comput 2:737–740
Wong ML, Leung KS (2000) Data mining using grammar based genetic programming and applications. Kluwer, Norwell