Applied Soft Computing 67 (2018) 558–566

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Diverse partner selection with brood recombination in genetic programming

Muhammad Waqar Aslam[a,*], Zhechen Zhu[b], Asoke Kumar Nandi[b,c]

[a] Department of Computer Systems Engineering, Mirpur University of Science and Technology (MUST), Mirpur 10250, AJK, Pakistan
[b] Department of Electronic & Computer Engineering, Brunel University, Uxbridge, Middlesex UB8 3PH, UK
[c] Department of Mathematical Information Technology, University of Jyväskylä, PO Box 35, Jyväskylä FI-40014, Finland

ARTICLE INFO

Article history:
Received 18 October 2017
Received in revised form 22 February 2018
Accepted 20 March 2018
Available online 27 March 2018

Keywords:
Genetic programming
Diversity
Partner selection
Brood recombination

ABSTRACT

The ultimate goal of learning algorithms is to find the best solution from a search space without testing each and every solution available in the search space. During the evolution process new solutions (children) are produced from existing solutions (parents), where new solutions are expected to be better than existing solutions. This paper presents a new parent selection method for the crossover operation in genetic programming. The idea is to promote crossover between two behaviourally (phenotype) diverse parents such that the probability of children being better than their parents increases. The relative phenotype strengths and weaknesses of pairs of parents are exploited to find out if their crossover is beneficial or not (diverse partner selection (DPS)). Based on the probable improvement in children compared to their parents, crossover is either allowed or disallowed. The parents qualifying for crossover through this process are expected to produce much better children and are allowed to produce more children than normal parents through brood recombination (BR). BR helps to explore the search space around diverse parents much more efficiently. Experimental results from different benchmarking problems demonstrate that the proposed method (DPS with BR) improves the performance of genetic programming significantly.

1. Introduction

Genetic programming (GP) has gained much attention in recent years because of its ability to produce human competitive solutions [1–3]. The flexibility in choosing various parameters of the algorithm and the ability to produce human interpretable solutions have made it superior to other learning algorithms and it has been applied quite frequently recently for solving real world problems. In [5] Liang, Zhang and Brownie used GP in image processing for figure ground segmentation. They demonstrated that GP based method was very successful in automatic segmentation of a variety of images. Maua and Grbac [6] used GP for software defect prediction for imbalanced datasets. Enrquez-Zrate et al. [7] used GP for predicting fuel flow and exhaust gas temperature of a gas turbine. A combination of GP and neuro-fuzzy methods was used for accurate prediction of suspended sediments in [8]. In another research GP was used for automatic scheduling of parallel unrelated machines [9]. Despite numerous advantages offered by GP there are some inherent issues in GP which limit its performance when applied to difficult tasks and there have been a variety of techniques proposed to improve its efficiency and performance. Elola et al. [10] exploited the predictive importance of intermediate solutions during a GP evolution and used it successfully for improving convergence rate. Genetic Programming in combination with fuzzy inference system was used for improving classification abilities of GP [11]. Ojha, Abraham and Snasel used diversity index measure for maintaining diversity in a multi objective GP and showed that it improved the efficiency of the proposed algorithm [13]. In [14], statistical GP using correlation based crossover and mutation was used to explore the search space more efficiently in less time. One of the main issues in GP is the premature convergence towards local optimum. It is widely accepted that the main reason for such convergence is the decrease in diversity in a fairly fit population of individuals as the population evolves. Ursem showed that a diversity guided evolutionary algorithm saved substantial amount of fitness evaluations and improved the performance remarkably [15]. Huang and Chen [16] used diversity pooling scheme for improving the convergence rate and showed that their proposed method reduced premature convergence rate. Various diversity measures were tested on standard problems for finding a relation between diversity and fitness in [17]. Important diversity measures for improving the search pro-

* Corresponding author.
E-mail addresses: waqar.cse@must.edu.pk (M.W. Aslam), zhechen.zhu@brunel.ac.uk (Z. Zhu), asoke.nandi@brunel.ac.uk (A.K. Nandi).

https://doi.org/10.1016/j.asoc.2018.03.035
1568-4946/© 2018 Elsevier B.V. All rights reserved.

cess were identified in this research. A diversity rewarding fitness function was used in [18] to avoid local optimum. Similar programs were replaced with randomly generated ones in [19,20] and it was shown that it minimised the occurrence of premature convergence. Fitness and solution diversity was increased and it was found that the high fitness solution were found more quickly in [21].

One of the main factors causing loss of diversity is the selection pressure imposed on the algorithm [22]. A high selection pressure would quickly fill up the population with fitter individuals because they have better chances of survival. These fitter individuals most likely would be similar to each other either in their structure or behaviour, resulting in a loss of diversity across the population. As a consequence, the algorithm may struggle to escape from a local optimum. Diversity in any population will help to avoid such local optimum.

Maintaining diversity is particularly important if the fitness landscape have many peaks and valleys resulting in many local optimums. In such a scenario, a more diverse population will increase the likelihood of GP in finding the optimal peak or relocating individuals in a dynamic landscape. However, despite such advantages, blind promotion of diversity may result in a loss of information previously gathered by the algorithm.

In this paper pairwise diversity is used to promote crossover between two diverse individuals. The method proposed in this study is a significant extension of the methods presented in [23,24]. In [23] Day and Nandi presented the idea of binary string representing strengths and weaknesses of an individual. Using the binary strings of different individuals comparative partner selection (CPS) was used to encourage crossover between two individuals having high pairwise diversity. In one of our earlier research [24] we suggested an improvement in CPS and used shared strengths of individuals in addition to high pairwise diversity while selecting a right partner for crossover. In this study we demonstrate that just having a high pairwise diversity or sharing same strengths may not necessarily lead to the optimum solution and within pairwise diversity we have introduced the concepts of *good diversity* and *bad diversity* to find the preferred partners for crossover. The novel concept of good and bad diversity helps to follow exploration and exploitation strategy, popular in machine learning algorithms. The preferred partners found using the proposed method are allowed to produce more children through brood recombination. Brood recombination helps to explore the search space close to diverse partners more efficiently which ultimately guides the process towards the optimum solution.

The paper is organised as follows: the diversity measures and role of diversity in guiding evolution is discussed in Section 2. Evaluation of binary string is discussed in Section 3. The comparative partner selection technique and its flaws are discussed in Section 4. The proposed method is discussed in Section 5. Experiments and results are discussed in Section 6, while conclusions are drawn in Section 7.

2. Genetic programming and diversity

In this section first we highlight different measures used in literature to calculate diversity and then the promotion of diversity in finding the optimum solution is discussed.

2.1. Diversity measures

Broadly, diversity represents the level and type of variety in a population of individuals. This variety could be in the form of structure (genotype diversity) or behaviour (phenotype diversity) of individuals. The genotype diversity measures the similarity between actual structures of individuals. The most common method for measuring this diversity is the edit distance which simply counts the number of node additions and deletions required to make any two individuals identical and gives a general idea about structural resemblance of any two individuals [25].

The phenotype diversity on the other hand represents the behaviour of individuals. The most common method for evaluating this diversity is to find the distribution of fitness values in a population [26]. In this method the fitness of an individual is calculated by dividing its standard fitness by the number of individuals sharing similar fitness values. Mckay introduced the idea of evaluating individual training cases for calculating fitness [27]. This idea was later used in [23] to promote crossover between two diverse individuals.

Some other measures based on both genotype and phenotype or occasionally using the combination of the two have also been proposed [28]. Ryan [29] presented the idea of evolving two different populations together with different fitness criteria where crossover was only allowed between two different populations. An increase in diversity with reduction in code bloat was reported. Genetic lineage strategies have also been proposed [30,31] where parents for crossover are selected based on their genetic lineage to promote diversity. Teller and Veloso proposed an internal reinforcement method for GP using neural learning [32]. The neural learning was used to update specific parts of GP programs as a function programs performance. Quang Uy et al. investigated the role of semantic locality of crossover in GP [33]. They defined a semantic distance metric for defining new crossover operators to improve semantic locality. They reported substantial advantage using semantic locality. Xie and Zhang investigated the selection of optimal crossover point [34] and reported that good crossover events have a strong preference for roots and bottoms of parent program trees.

Based on the above discussion it can be concluded that while various strategies have been proposed for promoting diversity, none have been widely adopted. Each strategy has its own advantages and disadvantages, and is suited for particular applications. The genotype diversity has the advantage that it is quite objective and two solutions having exactly same structure will have exactly same behaviour. This measure, however, does not consider the behavioural differences of individuals and only implies that the actual structures of individuals are not identical. While this method is widely adopted because of its simplicity, any two individuals categorised as structurally unique by this method may behave similarly because of the presence of introns and symmetric functions. On the other hand phenotype diversity is more subjective and it is mainly calculated using fitness values. A phenotype diversity based just on fitness does not give insight into actual behaviour of the population for individual cases of a test problem. Two solutions performing well on two different cases of the same problem will be categorised as similar.

Therefore, there is a need to find a measure which is inexpensive, informative and can relate diversity to fitness improvement. In this study a phenotype based strategy for controlling diversity and guiding the search towards the optimum solution is proposed.

2.2. Promotion of diversity

Most diversity measures give an overall indication of diversity of a population but they do not give any information about the quality of individuals present in the population. It is possible to maintain a high level of diversity without getting any benefit in the quality of individuals. Moreover, the level of diversity does not indicate whether a population has suffered sub-optimal convergence or not, because diversity does not always promote optimal convergence. While in some cases it has been shown that diversity avoids early or premature convergence, it is not beneficial all the time

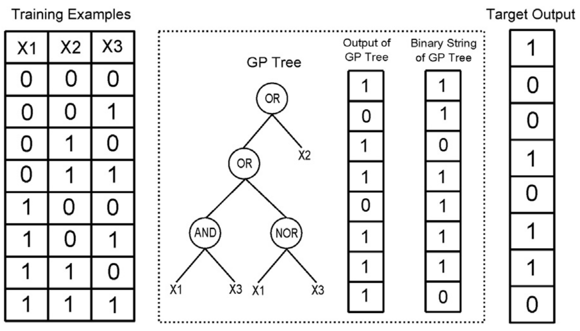

Fig. 1. An example of a GP Tree in a sub-optimally converged evolution for 3 bit parity problem.

and promoting it blindly can degrade performance. For example, if the population is very close to the optimum solution, maintaining diversity may slow down the rate of convergence.

A widely accepted dynamics for accurate search in most evolutionary methods is *exploration* and *exploitation*. The same technique has been used in this study using pairwise diversity. Initially a high value of diversity is maintained (*exploration*) called global search where GP is allowed to find globally the individuals that perform well. As the population becomes fitter and fitter, the level of diversity is decreased (*exploitation*) to do a local search. During local search, small changes are made through crossover to do a more focused search close to the optimum solution. Moreover, in the proposed method if there is no improvement at *exploitation* stage, the probability of mutation is increased to avoid a possible local optimum.

3. Binary string fitness characterisation

The idea of binary string fitness characterisation (BSFC) was introduced by Day and Nandi [23] which describes the efficacy of an individual in solving a given problem. Generally, a number of training cases are given to GP for finding the optimum individual which solves the given problem. The performance of any GP produced individual for solving all training cases is summed up in one parameter, *the fitness function*. A typical fitness function is the sum of errors for all training cases. Representing the ability of individuals by just one parameter is somewhat limiting as it does not give insight into response of individuals for each training case. Two individuals solving different but the same number of training cases will get the same overall fitness value and will be categorised as same. Although the fitness value is important in guiding the search towards the optimum individual, the response of individuals for each training case may help to explore the search space more efficiently.

In BSFC, the response of an individual for solving each training case is evaluated and a binary string (b_i) is created for each individual for storing that response. For logical problems where the output is either one or zero and is generally already known, the assignment of b_i is straight forward. If an individual solves a particular training case it will get a ‘1’ in the corresponding b_i bit, otherwise ‘0’. A ‘1’ is considered as strength and a ‘0’ is considered as weakness. An ideal or the optimum individual will have a binary string consisting solely of ‘1’s.

An example of a binary string for 3-bit parity problem in a sub-optimally converged run is shown in Fig. 1. On the left side of the figure are the training examples given to GP and on the right side of the figure are the target values. The variables X1, X2 and X3 represent the training examples. GP is asked to find an individual which can generate the target output from training examples. In the middle of the figure (inside dotted rectangular box) is an individual (tree) produced by GP to solve this problem in a sub-optimally converged run. It is sub-optimal as the individual produced by GP does not solve the problem completely. On the right side of the individual is the output produced by this individual which can be used to fill up the binary string attached to this individual. The binary string is filled up by comparing the output of the individual with target output.

The assignment of b_i is not as straightforward for non-binary problems. In [23], a method based on the average error for all training cases was proposed. If the error for a particular training case was less than the average error for all training cases, a ‘1’ was placed in its corresponding bit of b_i otherwise ‘0’. For classification problems a method based on the mean of output distribution was proposed in one of our earlier research [35]. Any input example within a certain standard deviations of mean of corresponding output distribution was considered as strength and an example away from the mean was considered as weakness.

4. Partner selection for crossover

Day and Nandi [23] used the binary string introduced in the previous section for partner selection during crossover operation and named it as comparative partner selection (CPS). In this section we outline the CPS method and discuss its limitations.

4.1. Comparative partner selection

In a standard genetic programming (SGP), the parents for crossover are chosen based on their fitness values while in CPS the parents are chosen based on their relative strengths and weaknesses. A crossover is encouraged if the two individual have strengths for different training examples and a crossover is discouraged if the two individuals share similar weaknesses. The idea behind this process is to reduce the phenotype variance and eradicate population weaknesses. The initial selection of parents in CPS is similar to SGP (chosen based on their fitness values); however in CPS, each pair of parents have to satisfy another criterion for crossover to take place. This criterion is based on simple logical operations and is used to encourage the crossover between two diverse (in terms of binary string) individuals. A crossover between two individuals is encouraged if one individual shows strength for an example for which the other individual shows weakness (*XOR*), while a crossover is discouraged if two individuals share similar weaknesses (*NOR*). The binary string can be used to calculate the probability of crossover (P_{cps}) between two individuals.

$$P_{cps}(b_1, b_2) = \frac{\sum XOR(b_1, b_2)}{\sum XOR(b_1, b_2) + \sum NOR(b_1, b_2)} \quad (1)$$

where P_{cps} is the probability of crossover, and b_1 and b_2 are the binary strings of two individuals. The probability of crossover will be high between two diverse (in terms of binary string) individuals compared to similar individuals.

The full process of CPS can be described as follows. Two parents p_1 and p_2 are selected based on their fitness values (individuals with higher fitness have more chances of being selected). These two parents (p_1 and p_2) are arranged in a way that p_1 is the fitter individual of the two (unless both have the same fitness). The P_{cps} is calculated between these two parents using Eq. (1). A random number is generated between 0 and 1 and if P_{cps} is greater than that random number, crossover takes place otherwise not. If crossover takes place then children might be added in new generation if they meet predefined selection criteria. If crossover does not take place, parent p_1 (the fitter individual) is kept and p_2 (the weaker/same strength) individual is dropped. A new second parent p_2 is selected using the same procedure and the above process is

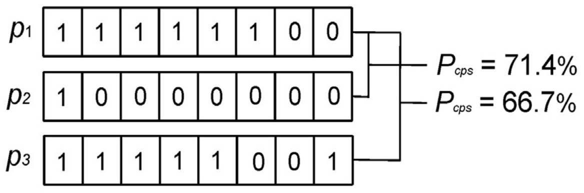

Fig. 2. An example showing limitation of CPS.

repeated to check if these two parents qualify for crossover. If a suitable partner is not found after $N/2$ attempts (N is population size), p_2 is chosen randomly ignoring CPS criterion. The CPS is penalised for not finding the right partner by decreasing the crossover rate by $1/N$ and increasing the mutation rate by $1/N$ to reintroduce diversity.

4.2. Limitations in comparative partner selection

Despite the obvious improvement in performance reported in [23], there are some limitations in CPS. This section highlights those limitations.

1. The aim of the CPS process is to reach the optimum solution through exploitation of individual's diversity but the way P_{cps} is calculated may actually not allow the process to move towards the optimum solution. An example is given in Fig. 2 to highlight this limitation. Fig. 2 shows the binary strings of three individuals. Suppose p_1 is our primary parent and it has to choose one of the other two parents (p_2 or p_3) for crossover. According to CPS criterion, the probability of crossover between p_1 and p_2 is 71.4%, while the probability of crossover between p_1 and p_3 is 66.7%. Let us first concentrate on crossover between p_1 and p_2, and see how much improvement this crossover can provide for evolution towards the optimum solution. Individual p_1 solves first six (left to right) training cases while individual p_2 solves only one. That means if a crossover between these two individuals takes place, their children are supposed to solve at best six training cases. So although the two parents are quite diverse, the children are less likely to have any improvement (*bad diversity*). In other words during this crossover, p_1 will share its strengths with p_2 but in terms of evolution towards the optimum solution, there is no benefit.

Now if a crossover takes place between p_1 and p_3, the example eight and six, which p_1 and p_3 were unable to solve respectively, might be solved by their children (*good diversity*). Although p_1 and p_2 are more diverse (5 bits different) while p_1 and p_3 are less diverse (2 bits different), the diversity between p_1 and p_3 is more important. So a crossover between p_1 and p_3 should have more probability if we want to evolve towards a better solution but CPS suggests exactly opposite.

2. After calculating the probability of crossover between two individuals, a random number is used to decide if they crossover or not. This may not be the best way to do it as some individuals having high probability of crossover may not actually have crossover while some individuals having low probability may qualify for crossover.

3. After two parents qualify for crossover satisfying CPS criterion, the following actions are taken. A node is randomly chosen on each parent and sub-branches downward from that node are swapped with each other. Since each parent can have many crossover points, choosing only one crossover point may not beget potentially better children. As the parents satisfying CPS criterion are expected to produce better children, search space close to these parents should be exploited more efficiently.

An example is given in Fig. 3 to demonstrate this fact. Fig. 3 shows two parents selected for crossover through CPS process for 3-bit parity problem with their respective binary strings. If we look at the binary strings of two parents, there are four examples at which the strength of one parent coincides with weakness of other

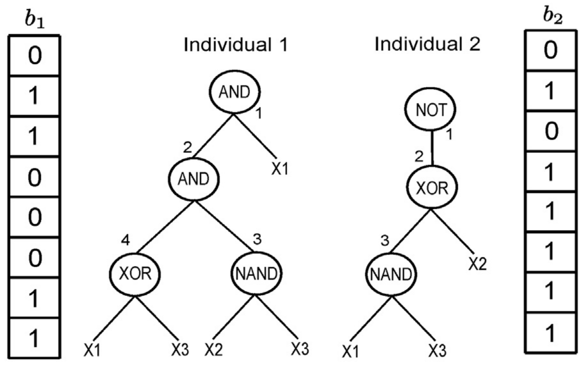

Fig. 3. Two individuals selected by CPS for crossover.

parent. According to CPS, the children produced by the crossover of these two parents could have strengths for these four examples. Let us suppose that a crossover takes place at node 3 of both parents. There will be two children produced by this crossover. The children when evaluated for 3-bit parity training examples will have these binary strings 01001011 and 01100011. None of these binary strings is better than parents' binary strings and there is no improvement in children compared to their parents. This shows us that a strength in one parent coinciding with a weakness of other parent may not result in a strength in children. The reason is that it is difficult to establish which part of an individual is responsible for certain strength and weakness.

Suppose a crossover takes place at node 4 (individual 1) and node 3 (individual 2). The binary strings of two children produced as a result of this crossover will be 01100011 and 11111111 respectively. This shows that this crossover produces a child which is the optimum solution and completely solves the problem. This example clearly shows that choosing only one point for crossover may ignore potentially good children. In this study we have used a technique based on brood recombination to solve this issue.

5. The proposed method

This section explains our proposed method. The method not only addresses the limitations in CPS method but also introduces some key parameters for guiding the search efficiently towards the optimum solution.

5.1. Diverse partner selection

One of the main limitations of CPS is the uncontrolled promotion of diversity. One very strong individual will be very diverse compared to very weak individual but a crossover between them may not help to achieve the ultimate goal. In such a case, a strong individual may share its strengths with a weak individual and a resulting child may not be any better. This will not help the evolution towards the optimum solution.

In our proposed method, randomly selected parents (based on their fitness values) are characterised using their binary strings. The better individual (in terms of binary string) is named as recipient and the weaker individual is named as donor. The cases solved by the donor ('1's) and not solved by the recipient ('0's) are the *good elements* and are the main focus for promoting crossover in our method. It is expected that bringing in elements of donor that could solve a case not solved by recipient would improve the fitness of the children. Simple logical operations are used to determine how many '1's of donor coincide with '0's of recipient. This number reflects the *good diversity* present between two parents and it is used to estimate the improvement a donor can provide to a recipient. If the improvement is significant, crossover takes place otherwise not. A question arises, how much improvement is a significant improvement? This question will be addressed later when we define our improvement parameters. The '1's of a recipient coinciding with '0's of a donor are used in CPS but not in our method as they represent the *bad diversity* present between parents. It is called *bad diversity* because recipient may share its strength with donor in this diversity and the resultant children may not be any better. The proposed method is named as diverse partner selection (DPS) and below are the steps of DPS.

Two parents are randomly selected based on their fitness values. They are named as recipient and donor based on their binary strings. The parent having more '1's is considered as recipient and the other one as donor. The following equations are used to find out if the two parents crossover. Let

$$Advantage = \sum AND(NOT(b_1), b_2) \tag{2}$$

where b_1 is the binary string of recipient and b_2 is the binary string of donor. The parameter *Advantage* gives the number of cases solved by a donor which are not solved by a recipient. Let

$$\alpha = \frac{Advantage}{\sum NOT(b_1)} * 100\% \tag{3}$$

The parameter α shows the percentage of cases solved by a donor that are not solved by a recipient. In other words it shows the possible improvement in the children of recipient and donor compared to recipient. Let

$$\beta = \left(1 - \frac{\sum b_1}{\sum(b_1 + NOT(b_1))}\right) * 100\% \tag{4}$$

The parameter β shows the room for improvement in recipient in percentage. It is important to note that the parameter α is calculated using only those bits for which recipient has '0's while the parameter β is calculated using all the bits of recipient. The values of parameters α and β are used to decide whether the possible improvement in children of recipient and donor is enough to have a crossover. Crossover between two parents is allowed only if α is greater than β. The motivation behind this condition is to use exploration and exploitation strategy which is very popular in machine learning field. The details about how this condition ensures exploration and exploitation are given later in this section.

In practice two parents p_1 and p_2 are selected based on their fitness values, and the values of α and β are calculated. If α is greater than β, crossover takes place, otherwise not. If crossover does not take place, a new second parent p_2 is selected without changing p_1 and the above procedure is repeated to find out if they can have a crossover. If a suitable p_2 is not found after $N/2$ operations, crossover rate is decreased by $1/N$ and mutation rate is increased by $1/N$ in the current generation (N is total population size). The crossover and mutation rates return back to their initial values in the next generation.

At the start of evolution a one-bit improvement provided by donor will result in a slight increase in α (because of abundance of zeros in recipient). As the population becomes fitter and fitter, and moves towards the optimum solution, the number of '1's in any recipient will increase and even a single '1' of donor coinciding with '0' of recipient will result in a significant increase in α. The trend in β is quite the opposite. Initially most of the individuals will be weak which means a high β value and as the population becomes fitter the β value will reduce.

Since at the start of evolution α will be low and β will be high, most individuals will fail to satisfy DPS criterion resulting in an increase in mutation rate (exploration). As the population becomes fitter, α will be high and β will be low which means more individuals will satisfy DPS criterion (exploitation through BR). So the proposed method will be able to maintain a high diversity at the start of evolution (through mutation) and more optimised search will be conducted as the population moves towards the optimum solution (through BR). However, if there is no improvement at exploitation stage, α will become low again causing mutation rate to increase to avoid local optimum.

Let us use the example given in Fig. 2 to elaborate further. Earlier we concluded that the probability of crossover between parents p_1 and p_3 should be higher compared to parents p_1 and p_2 but CPS suggested opposite. Let us see the probabilities of crossover in DPS. The values of α and β for the parents p_1 and p_2 are 0% and 25% respectively, while the values for the parents p_1 and p_3 are 50% and 25% respectively. So in DPS p_1 and p_2 do not qualify for crossover, while p_1 and p_3 qualify for crossover which should guide the process towards the optimum solution as discussed in Section 4.2.

5.2. Brood recombination

We demonstrated using an example in limitation number 3 of CPS method (Fig. 3) that choosing only one point for crossover may neglect potentially good children. In this study to give a fair chance to the parents selected through DPS process, they are allowed to produce more children than normal parents. Each pair of parents is allowed to have five crossovers (producing ten children), each time with a different crossover point. The best two children out of ten are selected while rest are discarded. This way the search space close to DPS selected parents which are supposed to have the ability to produce much better children is exploited more efficiently. This idea of allowing certain parents to produce more children was first proposed by Tackett and is known as brood recombination [36].

So the proposed method have two major difference from the standard genetic programming algorithm. First, parents are passed through the DPS process and second, the parents satisfying the DPS criterion are given the facility of brood recombination.

6. Experiments and results

Six benchmarking problems are considered in this study. These problems come from three different problem domains: logical problems (3-bit parity, 5-bit parity, 11-bit multiplexer), regression problem (quartic polynomial), and classification problems ("ionosphere" and "spect heart" data classification). The parameters used for all the problems are given in Table 1. The fitness function used for logical and regression problems was the sum of the errors, while the classification accuracy was used as fitness function for classification problems. In order to see the effect of DPS and BR separately, five different kinds of GP combinations/methods are used for experiments: the standard GP (SGP), CPS, DPS, CPS with BR (CPS-BR), and DPS with BR (DPS-BR); in CPS-BR the parents satisfying CPS criterion were given the facility of BR.

6.1. Parity problems

Parity problems have often been used as benchmarking problems in GP [37]. These problems can be divided as even parity or odd parity. For even parity problems, if there are even number of ones in input training string, the output is one else zero. Similarly for odd parity problems, if the number of ones in input string is odd, the output is one otherwise zero. In this research, even parity problems are used. The function pool used for the 3-bit and 5-bit problems are different. Since the 3-bit problem is simpler than the 5-bit, a simple function pool is used for this problem while a richer function pool is used for 5-bit problem. The parity problem is a highly non-linear problem in terms of searching its solution as demonstrated in [38] and most hill climbing algorithms fail to find solution of this prob-

Table 1
Parameters used for the experimental work.

Parameter	Standard value
Generations (for all problems)	100
Population size	
3-bit parity	100
5-bit parity	100
Multiplexer	100
Regression	200
Classification	50
Function pool	
3-bit parity	{AND, OR, NAND, NOR}
5-bit parity	{AND, OR, NAND, NOR, XOR}
Multiplexer	{AND, OR, NOT, IF}
Regression	{+, −, x, sin, cos, [*]mylog}
Classification	{+, −, x, reciprocal, negator, abs, sqrt, sin, cos, tan, asin, acos, tanh, [*]mylog}
Terminal pool	
3-bit parity	{B1, B2, B3}
5-bit parity	{B1, B2, B3, B4, B5}
Multiplexer	{B1, B2, . . ., B11}
Regression	X1
Classification	Input features
Operators	Crossover, Mutation
Operator probabilities	(60%, 40%)
Tree generation	Ramped half and half
Initial maximum depth	4
Maximum depth	17
Selection operator	Roulette
Elitism	half-elitism

[*] Mylog is a protected $log_e(x)$ function which ignores input if it is zero.

lem consistently. The b_i for the 3-bit parity problem will have 8 bits, while for the 5-bit parity problem it will be 32 bits long.

6.1.1. 3-bit parity

The experiment was performed using the parameters given in Table 1. Each experiment for each GP method (SGP, CPS, DPS, CPS-BR, DPS-BR) was performed 100 times and the performance was averaged over 100 runs. Since the fitness is the sum of errors, the lower the fitness the better it is. Fig. 4 shows the comparison between all the methods in terms of fitness value averaged over 100 runs. The SGP performed the worst out of all the methods followed by CPS. The performance of CPS-BR was almost similar to DPS for most of the generations and there was not much difference in performance between the two. This demonstrates the superiority of DPS as without BR it was able to achieve a performance similar to CPS-BR. The DPS-BR outperformed all the other methods comprehensively by a big margin. The average fitness values and standard deviations at 100th generation for SGP, CPS, DPS, CPS-BR and DPS-BR were 0.58, 0.25, 0.09, 0.1, 0.0 and 0.62, 0.44, 0.31, 0.29, 0.0 respectively. The number of runs successful in finding the optimum solution were 49, 75, 90, 91, and 100 (out of 100) for SGP, CPS, DPS, CPS-BR and DPS-BR respectively. These numbers clearly show that DPS-BR is more accurate and robust compared to all other methods.

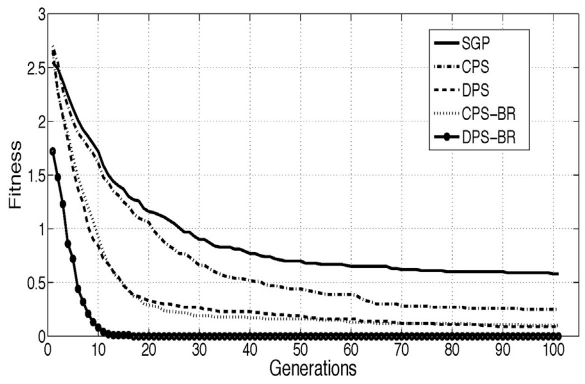

Fig. 4. Average fitness of best of generation for 100 runs for 3-bit parity problem.

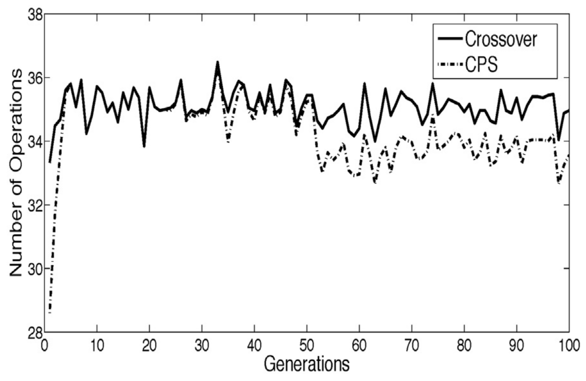

Fig. 5. Crossover and CPS comparison for sub-optimal runs.

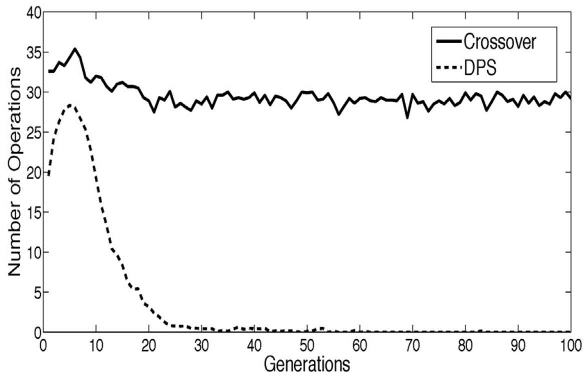

Fig. 6. Crossover and DPS comparison for sub-optimal runs.

In any generation there would be a number of crossover operations between different partners out of which some will satisfy CPS/DPS criterion and some will not. The purpose of CPS/DPS criterion is to use the diversity in the population and ultimately to increase the rate of optimal convergence. If a GP run is unsuccessful or converges sub-optimally that should mean the diversity is low and should be reflected by number parents satisfying CPS or DPS criterion. In order to find this out, a comparison between total crossover operations and the crossover operations satisfying CPS/DPS criterion is given in Figs. 5 and 6. Figures show the total number of crossover operations and the operations satisfying CPS/DPS criterion at each generation for sub-optimal runs.

Fig. 5 shows that for CPS method, the number of parents satisfying CPS criterion was very close to total crossover operations (i.e. the diversity was high), although all these runs resulted in sub-optimal convergence (not shown in Figure). This demonstrates that the diversity measures used by CPS were not able to guide the process towards the optimum solution despite the satisfaction of a lot of partners for CPS criterion. On the other hand, for DPS for sub-optimal runs, the number of parents satisfying DPS criterion was low compared to the total crossovers (Fig. 6), which means the diversity was low for these runs. This shows that the diversity measures used by DPS criterion correctly showed the lack of diversity for sub-optimal runs.

6.1.2. 5-bit parity

Similar experiments were conducted for the 5-bit parity problem using the parameters given in Table 1. The fitness value averaged over 100 runs is shown in Fig. 7. The average fitness val-

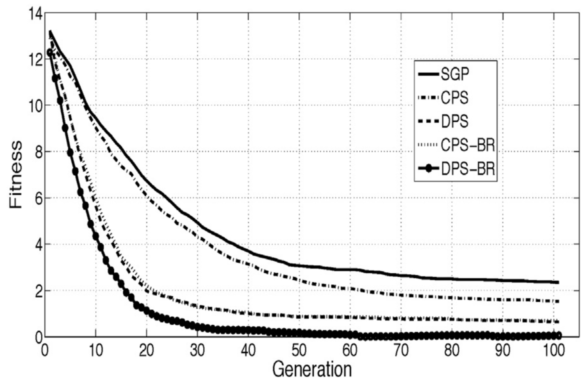

Fig. 7. Average fitness of best of generation for 100 runs for 5-bit parity problem.

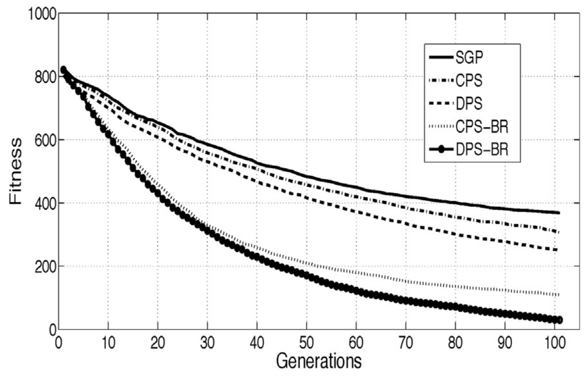

Fig. 8. Average fitness of best of generation for 100 runs for multiplexer problem.

ues and standard deviations at 100th generation for SGP, CPS, DPS, CPS-BR and DPS-BR were 2.35, 1.50, 0.70, 0.68, 0.05 and 2.72, 2.10, 1.40, 1.38, 0.76 respectively. The number of runs successful in finding the optimum solution were 44, 54, 74, 75, 84 (out of 100) for SGP, CPS, DPS, CPS-BR, and DPS-BR respectively. These results are consistent with the 3-bit parity problem, and again SGP and CPS have the poorest performance. The performance of DPS and CPS-BR are again very similar while the performance of DPS-BR is the best.

6.2. *The multiplexer problem*

The multiplexer problem has also been used as a benchmarking problem in the past [37]. It is an 11-bit problem, so the total number of input combinations are 2048. The function pool and other parameters used are given in Table 1. The performance curves for this problem are shown in Fig. 8 which again shows the superiority of DPS-BR method. SGP was not able to find any optimum solution in all 100 runs while CPS and DPS found 1 and 23 (out of 100) optimum solutions respectively. On the other hand CPS-BR and DPS-BR found 71 and 80 optimum solutions respectively.

6.3. *Quartic polynomial problem*

It is a symbolic regression problem where GP tries to regress to a curve given by the following equation.

$$f(x) = x^4 + x^3 + x^2 + x \tag{5}$$

where x is the input variable and $f(x)$ is the target function. The number of training cases used are 21 where the domain of x is from -1 to $+1$ with an increment of 0.1. The binary string for this problem is calculated using average error technique proposed in [23]. The results for this problem are shown in Fig. 9. It is clear from the

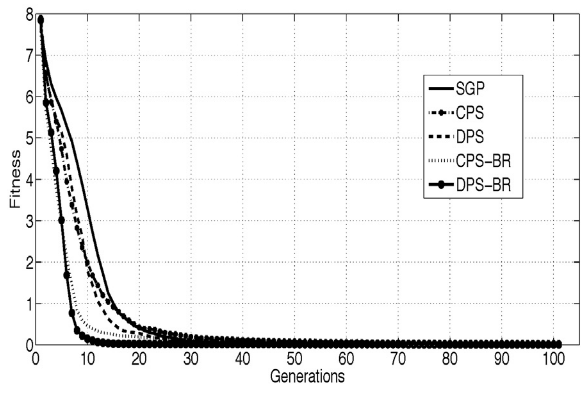

Fig. 9. Average fitness of best of generation for 100 runs for quartic polynomial problem.

figure that the performance of the proposed method is superior to other methods. The number of optimum solutions found by SGP, CPS, DPS, CPS-BR, and DPS-BR were 78, 79, 83, 88 and 94 (out of 100) respectively.

6.4. *Classification problems*

The use of GP for classification problems is not new and it has been used for classification in various domains in the past [39–42]. Two classification datasets are used in this study which are taken from UCI Machine Learning Repository [43]. The problems used in this study are ”ionosphere“ and ”spect heart“, which belong to binary classification problems. The classification accuracy used as fitness function is given below

$$\text{Accuracy} = 0.5 * (\text{Sensitivity} + \text{Specificity}) \tag{6}$$

where the sensitivity and specificity were calculated using the method given in [44] which uses Gaussian distribution model to find the optimum threshold separating any two classes. The binary string for classification problems has been evaluated using the method given in an earlier study [35]. Any training sample gets a ’1‘ in the binary string if it is within half the standard deviation of mean of its corresponding class distribution otherwise ’0‘. The function pool used for classification problems is given in Table 1.

6.4.1. *“Ionosphere” data classification*

This dataset contains 351 radar signals out of which 126 (35.8%) are *good* signals and 225 (64.2%) are *bad* signals. Since the dataset is imbalanced, it is divided into training and test datasets such that balanced data is used for training to avoid evolution towards biased classifiers. The number of training samples used were 92 for each class. The fitness curves for this problem are shown in Fig. 10. It is clear from the figure that DPS-BR outperforms the other methods. The trend in performance is similar to benchmarking problems where SGP has the worst performance followed by CPS, DPS, CPS-BR, and DPS-BR. The classification accuracies for both training and test data along with standard deviations are given in Table 2, demonstrating the advantages of the proposed method for both training and test datasets.

6.4.2. *“Spect heart” data classification*

There were 267 samples in this dataset out of which 55 (20.6%) samples are abnormal and 212 (79.4%) are normal. Again the dataset is divided to make the training data balanced. The training samples used were 40 for each class. The fitness curves are given in Fig. 11. Again the proposed method performs better than the other methods and all the classification results are consistent with previous results. The classification results for training and test

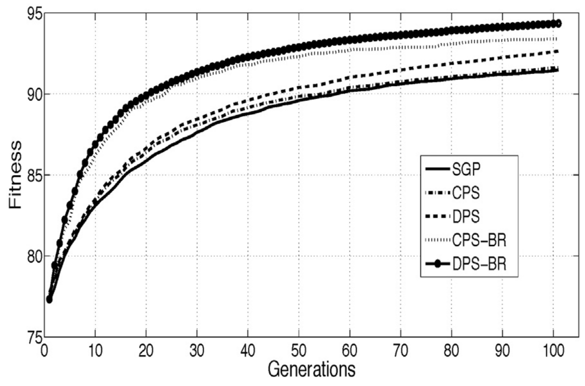

Fig. 10. Average fitness of best of generation for 100 runs for “ionosphere” problem.

Table 2
Classification results.

Dataset	Method	Training	Testing
“Ionosphere”	SGP	91.5% ± 2.3	91.7% ± 3.9
	CPS	91.6% ± 2.1	91.8% ± 3.1
	DPS	92.6% ± 1.9	92.0% ± 3.1
	CPS-BR	93.4% ± 1.9	92.3% ± 3.0
	DPS-BR	94.4% ± 1.8	92.4% ± 3.0
”Spect Heart“	SGP	85.6% ± 2.6	72.9% ± 3.7
	CPS	86.8% ± 2.5	74.0% ± 3.4
	DPS	88.2% ± 2.3	74.3% ± 3.3
	CPS-BR	88.8% ± 2.2	74.5% ± 3.3
	DPS-BR	89.8% ± 2.0	74.9% ± 3.1

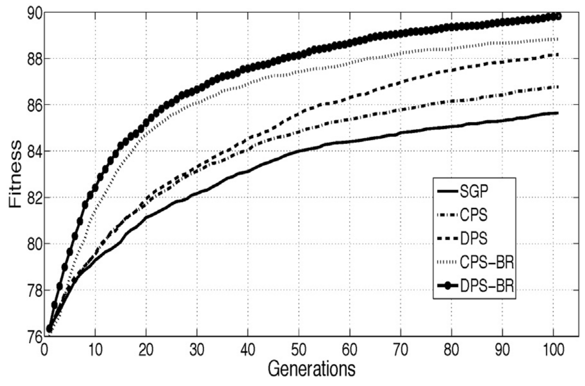

Fig. 11. Average fitness of best of generation for 100 runs for ”spect heart“ problem.

datasets are given in Table 2, indicating the superiority of the proposed method. The classification accuracies for test dataset is low due to highly imbalanced nature of test dataset.

Based on all the above discussions it can be concluded that the proposed method (DPS-BR) performs better than all the other methods for a variety of problems. Experiments reveal that the improvement shown by the proposed method is not limited to benchmarking problems and it outperforms other methods for real world classification problems as well.

The DPS method uses simple logical operations for finding the probability of crossover and its computational cost is similar to CPS. The BR method introduces some extra computation for both CPS and DPS but the performance of DPS-BR is much superior to CPS-BR and the additional computations cost added by BR is outweighed by the significant increase in performance. In fact, for benchmarking problems DPS-BR is much faster compared to the other methods since it finds the optimum solution very quickly. It is on average 10 times faster than SGP (based on average time taken to find the optimum solution). For classification problems as there is no optimum solution, the cost of DPS-BR is more than the SGP, CPS, DPS, and is similar to CPS-BR. SGP is almost two times faster than DPS-BR for classification problems. This additional cost of DPS-BR is not huge if we consider the performance improvement shown by this method.

7. Conclusion

This study has proposed a novel method (DPS-BR) for promoting crossover between behaviourally diverse individuals with the aim to guide evolution towards the optimum solution. The pairwise diversity of parents is divided into good and bad diversity based on its ability to improve the performance of children (DPS). The parents with good diversity are allowed to produce more children to exploit the search space closer to them more efficiently (BR). DPS exploits the good diversity present in the population during crossover operation and increases the mutation rate if there is not enough good diversity across a population. The method has built-in ability of exploration at the start of evolution and exploitation at the end of evolution. Moreover, it can also switch from an exploitation stage to an exploration stage if caught in a local optimum.

The proposed method when applied on benchmarking problems not only outperformed other methods in terms of fitness value but also increased the probability of finding an optimum solution significantly. The results were similar when the proposed method was applied on real world classification problems where it showed better classification accuracy compared to the other methods.

Acknowledgements

The authors thank the UCI Machine Learning Repository for the datasets. Zhechen Zhu would like to thank the Department of Electronic and Computer Engineering, Brunel University, for their financial support. Asoke Nandi would like to thank TEKES for the award of the Finland Distinguished Professorship.

References

[1] J.R. Koza, Human-competitive machine invention by means of genetic programming, Artif. Intell. Eng. Des. Anal. Manuf. 22 (3) (2008) 185–193.
[2] H. Iba, Y. Hasegawa, T.K. Paul, Applied Genetic Programming and Machine Learning, 1st ed., CRC Press, Inc, 2009.
[3] R. Poli, W.B. Langdon, N.F. McPhee, A Field Guide to Genetic Programming, Lulu Enterprises, UK Ltd, 2008.
[5] Y. Liang, M. Zhang, W.N. Browne, Genetic programming for evolving figure-ground segmentors from multiple features, Appl. Soft Comput. 51 (2017) 83–95.
[6] G. Mauša, T.G. Grbac, Co-evolutionary multi-population genetic programming for classification in software defect prediction: an empirical case study, Appl. Soft Comput. 55 (2017) 331–351.
[7] J. Enríquez-Zárate, L. Trujillo, S. de Lara, M. Castelli, E. Z-Flores, L. Muñoz, A. Popovič, Automatic modeling of a gas turbine using genetic programming: an experimental study, Appl. Soft Comput. 50 (2017) 212–222.
[8] E. Shamaei, M. Kaedi, Suspended sediment concentration estimation by stacking the genetic programming and neuro-fuzzy predictions, Appl. Soft Comput. 45 (2016) 187–196.
[9] M. Durasevic, D. Jakobovic, K. Knezevic, Adaptive scheduling on unrelated machines with genetic programming, Appl. Soft Comput. 48 (2016) 419–430.
[10] A. Elola, J.D. Ser, M.N. Bilbao, C. Perfecto, E. Alexandre, S. Salcedo-Sanz, Hybridizing Cartesian genetic programming and harmony search for adaptive feature construction in supervised learning problems, Appl. Soft Comput. 52 (2017) 760–770.
[11] A.S. Koshiyama, M.M. Vellasco, R. Tanscheit, GPFIS-CLASS: a genetic fuzzy system based on genetic programming for classification problems, Appl. Soft Comput. 37 (2015) 561–571.
[13] V.K. Ojha, A. Abraham, V. Snášel, Ensemble of heterogeneous flexible neural trees using multiobjective genetic programming, Appl. Soft Comput. 52 (2017) 909–924.
[14] M.A. Haeri, M.M. Ebadzadeh, G. Folino, Statistical genetic programming for symbolic regression, Appl. Soft Comput. 60 (2017) 447–469.
[15] R.K. Ursem, Diversity-guided evolutionary algorithms, in: Proceedings of the Congress on Evolutionary Computation, IEEE Press, 2002, pp. 1633–1640.
[16] T.Y. Huang, Y.Y. Chen, Diversity-based selection pooling scheme in evolution strategies, Proceedings of the ACM symposium on Applied computing (2001) 351–355.

[17] E. Burke, S. Gustafson, G. Kendall, Diversity in genetic programming: an analysis of measures and correlation with fitness, IEEE Trans. Evolut. Comput. 8 (1) (2004) 47–62.
[18] T.F. Bersano-Begey, Controlling exploration, diversity and escaping local optima in GP: adapting weights of training sets to model resource consumption, Late Breaking Papers at the 1997 Genetic Programming Conference (1997) 7–10.
[19] V. Ciesielski, D. Mawhinney, Prevention of early convergence in genetic programming by replacement of similar programs, Proceedings of the 2002 Congress on Evolutionary Computation, vol. 1 (2002) 67–72.
[20] D. Mawhinney, Preventing early convergence in genetic programming by replacing similar programs, Proceedings of the Congress On Evolutionary Computation (2000) 67–72.
[21] G. Chen, C.P. Low, Z. Yang, Preserving and exploiting genetic diversity in evolutionary programming algorithms, IEEE Trans. Evolut. Comput. 13 (3) (2009) 661–673.
[22] H. Xie, M. Zhang, Parent selection pressure auto-tuning for tournament selection in genetic programming, IEEE Trans. Evolut. Comput. 17 (1) (2013) 1–19.
[23] P. Day, A.K. Nandi, Binary string fitness characterization and comparative partner selection in genetic programming, IEEE Trans. Evolut. Comput. 12 (6) (2008) 724–735.
[24] M.W. Aslam, Z. Zhu, A.K. Nandi, Improved comparative partner selection with brood recombination for genetic programming, in: IEEE International Workshop on Machine Learning for Signal Processing (MLSP), IEEE, Southampton, UK, 2013.
[25] U.M. O'Reilly, Using a distance metric on genetic programs to understand genetic operators, IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation, vol. 5 (1997) 4092–4097.
[26] D.E. Goldberg, J. Richardson, Genetic algorithms with sharing for multimodal function optimization, Proceedings of the Second International Conference on Genetic Algorithms and their application (1987) 41–49.
[27] R.I. Mckay, Fitness sharing in genetic programming, in: Proceedings of the Genetic and Evolutionary Computation Conference, Morgan Kaufmann, 2000, pp. 10–12.
[28] J. Rosca, Entropy-driven adaptive representation, in: Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications, Morgan Kaufmann, 1995, pp. 23–32.
[29] C. Ryan, Advances in Genetic Programming, Chap, Pygmies and Civil Servants, MIT Press, 1994, pp. 243–263.
[30] E.K. Burke, S. Gustafson, G. Kendall, N. Krasnogor, Is increased diversity in genetic programming beneficial? An analysis of the effects on performance, in: Proceedings of the 2003 Congress on Evolutionary Computation, IEEE Press, 2003, pp. 1398–1405.
[31] N.F. Mcphee, Analysis of genetic diversity through population history, in: Proceedings of the Genetic and Evolutionary Computation Conference, Morgan Kaufmann, 1999, pp. 1112–1120.
[32] A. Teller, M. Veloso, Internal reinforcement in a connectionist genetic programming approach, Artif. Intell. 120 (2) (2000) 165–198.
[33] N.Q. Uy, N.X. Hoai, M. O'Neill, R. McKay, D.N. Phong, On the roles of semantic locality of crossover in genetic programming, Inf. Sci. 235 (0) (2013) 195–213.
[34] H. Xie, M. Zhang, Depth-control strategies for crossover in tree-based genetic programming, Soft Comput. 15 (9) (2011) 1865–1878.
[35] M.W. Aslam, A.K. Nandi, Detection of diabetes using genetic programming, Proceedings of the 18th European Signal Processing Conference (EUSIPCO) (2010) 1184–1188.
[36] W.A. Tackett, Recombination, Selection and the Genetic Construction of Computer Programs, Ph. D. thesis, University of Southern California, Department of Electrical Engineering Systems, 1994.
[37] J.R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press, 1992.
[38] W. Banzhaf, P. Nordin, R.E. Keller, F.D. Francone, Genetic Programming: An Introduction, Morgan Kaufmann, San Mateo, CA, 1998.
[39] D.P. Muni, N.R. Pal, J. Das, A novel approach to design classifiers using genetic programming, IEEE Trans. Evolut. Comput. 8 (2) (2004) 183–196.
[40] P. Espejo, S. Ventura, F. Herrera, A survey on the application of genetic programming to classification, IEEE Trans. Syst. Man Cybern. C: Appl. Rev. 40 (2) (2010) 121–144.
[41] M.W. Aslam, Z. Zhu, A.K. Nandi, Automatic modulation classification using combination of genetic programming and KNN, IEEE Trans. Wirel. Commun. 11 (8) (2012) 2742–2750.
[42] U. Bhowan, M. Johnston, M. Zhang, Developing new fitness functions in genetic programming for classification with unbalanced data, IEEE Trans. Syst. Man Cybern. B: Cybern. 42 (2) (2012) 406–421.
[43] A. Frank, A. Asuncion, UCI Machine Learning Repository, 2010.
[44] M. Zhang, W. Smart, Using Gaussian distribution to construct fitness functions in genetic programming for multiclass object classification, Pattern Recogn. Lett. 27 (2006) 1266–1274.