Computers in Human Behavior xxx (2010) xxx–xxx

Contents lists available at ScienceDirect

Computers in Human Behavior

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

DepthLimited crossover in GP for classifier evolution

Hajira Jabeen *, Abdul Rauf Baig

Department of Computer Science, National University of Computer and Emerging Sciences, Islamabad, Pakistan

ARTICLE INFO

Article history:
Available online xxxx

Keywords:
Genetic Programming
Crossover
DepthLimited
Bloat
Classification
Data mining

ABSTRACT

Genetic Programming (GP) provides a novel way of classification with key features like transparency, flexibility and versatility. Presence of these properties makes GP a powerful tool for classifier evolution. However, GP suffers from code bloat, which is highly undesirable in case of classifier evolution. In this paper, we have proposed an operator named "DepthLimited crossover". The proposed crossover does not let trees increase in complexity while maintaining diversity and efficient search during evolution. We have compared performance of traditional GP with DepthLimited crossover GP, on data classification problems and found that DepthLimited crossover technique provides compatible results without expanding the search space beyond initial limits. The proposed technique is found efficient in terms of classification accuracy, reduced complexity of population and simplicity of evolved classifiers.

1. Introduction

Genetic Programming (GP) introduced by Koza (1992) is used to automatically construct and evolve computer programs. This new flexible and interesting technique has solved numerous interesting applications. One of these applications is data classification. The need of data classification arises for the tasks like face recognition, speech recognition, fraud detection and knowledge extraction from databases etc.

Data classification can be defined as assigning class labels to a data instance based upon knowledge gained from previously seen labeled data. Various classification algorithms have been proposed and used for their simplicity, understandability or accuracy. Simpler techniques like decision trees are simple and understandable but applicable to small data sets only. Whereas statistical techniques or neural networks require expertise and the results are not comprehensible.

GP offers a distinctive method of representing classifiers in the form of "Numeric Expression Trees" (Loveard & Ciesielski, 2001), or "Genetic Programming Classifier Expressions" (Kishore, Patnaik, Mani, & Agrawal, 2000). These expression trees are evolved using terminal set containing attribute values and an ephemeral constant, and function set containing arithmetic operators +, −, ∗ and /. For each data instance the tree outputs a real integer. This value is mapped to a class. This value-to-class mapping is simple in case of binary classification where positive value corresponds to one class and negative value to the other. The challenge arises in case of multi-class classification, where a single real output is to be mapped to more than two classes. Several methods have been proposed in this regard. One of these methods assigns different thresholds for different classes, this includes static thresholds (Loveard & Ciesielski, 2001; Yang & Smart, 2004), dynamic thresholds (Yang & Smart, 2004) and slotted thresholds (Yang & Smart, 2004). Another method is binary decomposition proposed by Kishore et al. (2000) and Loveard & Ciesielski (2001). A multi-class problem is decomposed into several binary problems and different classifiers are evolved for each class. The result is as many classifiers as there are classes in the data. In this method only one classifier is supposed to output a positive value, or a conflict arise, for which different conflict resolution methods are present in the literature (Kishore et al., 2000; Muni, Pal, & Das, 2004).

GP is an evolutionary algorithm using the recombination operators like crossover and mutation. Crossover is the method of swapping two subtrees from two parent trees and mutation randomly changes a part of a selected tree. Both these operators play an important role to search through the hyperspace of desired solutions and maintain genetic diversity in GP.

The GP based classification has several advantages over other classification techniques. The GP evolved classifiers are transparent and clearly show the relationship among attributes of data in form of a mathematical equation (Kishore et al., 2000) or rules (Engelbrecht, Rouwhorst, & Scheoman, 2002) depending upon the type of data and primitive set used for classifier evolution. The size and structure of a GP tree is not fixed, which offers a flexible search space to probe for the best classification rule. GP has freedom of expression in every way, until it adds to the classification accuracy and adheres to the initial syntax defined for tree structure. GP is readily applicable to the data in its original form and no transformation of data is required (Kishore et al., 2000). GP has an inherent ability to model the data dependence in its evolving structures

* Corresponding author. Tel.: +92 321 5708908.
E-mail address: hajira.jabeen@nu.edu.pk (H. Jabeen).

0747-5632/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.chb.2010.10.011

Please cite this article in press as: Jabeen, H., & Baig, A. R. DepthLimited crossover in GP for classifier evolution. *Computers in Human Behavior* (2010), doi:10.1016/j.chb.2010.10.011

without any explicit knowledge of underlying structure of dependencies. GP can eliminate attributes unnecessary for classification task, discarding the need of explicit feature extraction algorithm (Kishore et al., 2000).

In addition to above mentioned benefits, GP has a well known drawback of increase in population complexity during evolution (Poli, Langdon, & McPhee, 2008). That is the average number of nodes in population starts increasing. This increase in tree size is also termed as code bloat or code growth. In case of classifier evolution, one is always interested in evolving simple and comprehensible rules (Carreno, Leguizamon, & Wagner, 2007). The increase in complexity of evolved expressions makes them less understandable. Larger trees tend to over-fit the training data decreasing the testing accuracy (Kotsiantis, 2007) whereas the population is evolved in anticipation of increase in accuracy over unseen data. Moreover, larger and complex population adds more computation time in an already computationally expensive technique. These factors raise the need of evolving simpler classifiers using lesser computation and time.

In this paper we propose a method to overcome this drawback of increase in complexity in terms of average number of nodes by limiting the depth of subtrees for crossover operation (DepthLimited crossover). "Depth of a node is number of edges that need to be traversed to reach the node starting from the tree's root node (which is assumed to be at depth 0). The depth of a tree is the depth of its deepest leaf" (Poli et al., 2008). The proposed method is efficient in terms of classification accuracy and it does not let the complexity of trees increase during the evolution. In this paper we have

- Presented a new crossover operator
- Eliminated bloat by limiting the search space
- Reduced the complexities of evolved classifiers
- Achieved compatible classification results

After a brief introduction in Section 1, Section 2 discusses work relevant to classification and different crossover techniques used in GP. Section 3 presents and explains the work proposed by authors. Obtained results are organized and discussed in Section 4. Section 5 concludes the findings and highlights possible future work.

2. Literature review

GP has been of interest for various researchers. The research directions covered in this part include use of GP for classification, reasons for occurrence of code bloat and proposition of crossover operators to reduce bloat or to favor efficient search.

2.1. GP for classification

GP has been an area of interest for various researchers during the previous years. It had been applied to solve various problems, one of those being data classification. Several methods have been proposed to tackle data classification. They can be categorized into three different types.

2.1.1. Evolution of classification algorithms

In some works (Pappa & Freitas, 2006) the grammar to GP evolution is evolved in such a way that ultimate result is a feasible classification algorithm. For algorithm evolution, GP uses some type of constrained syntax/grammar so that the trees can transform into an algorithm and remain valid after application of evolutionary operators. Other methods to evolve classification algorithms like decision trees (Eggermont, 2005) and fuzzy decision trees have also been presented.

2.1.2. Evolution of classification rules

Evolution of classification rules have been proposed by various researchers (Carreno et al., 2007; Engelbrecht et al., 2002; Johansson, Niklasson, & König, 2007). In this type of methods GP trees are evolved as logical rules. These methods are usually applied to categorical, nominal or mixed data.

2.1.3. Evolution of classification expressions

A newer and somewhat GP specific method is evolution of classifiers in the form of mathematical expressions (Kishore et al., 2000; Muni et al., 2004; Yang & Smart, 2004). In this type of classification the classifiers are evolved in the form mathematical discriminating expressions. A classifier expression is created using numerical attributes of the data and some random constants. The value of expression is evaluated for every instance of the data where the output is a real value. This real output is mapped onto different classes of the data.

2.1.4. Multi-class classification

One of the simple methods of using classifier expressions for multiclass classification is binary decomposition proposed by Kishore et al. (2000) and Loveard and Ciesielski (2001) .One expression is evolved for each class present in the dataset considering all the other classes as reject class using classification accuracy as fitness measure. The cumulative result of expressions for each class is used to determine the final classification decision. A relatively different approach is used by (Smart & Zhang, 2005), in which discriminating distances are also used as a fitness measure and evolved classifiers in one evolutionary cycle, another method used by (Yang & Smart, 2004) is using different thresholds (boundaries) for different classes, various thresholds mechanisms (static, dynamic, slotted) have been presented. A GA inspired approach is presented by Muni et al. (2004) where classifiers for each class are evolved as a chromosome containing discriminating expressions for each class. Fitness is classification accuracy of whole chromosome.

None of the above methods use any explicit mechanism to avoid code bloat, limits to max tree size (Muni et al., 2004), or max tree depth (Smart & Zhang, 2005) are applied avoid increase in tree complexities.

2.2. Code growth

The problem of code growth is well reported in GP literature. It is important to eliminate this tendency because code bloat not only makes the process of evolution computationally expensive but also reduces the understandability and generalization ability of evolved classifiers (Poli et al., 2008).

Different interesting theories for the occurrence of code bloat exist. An interesting reason presented by Langdon (Langdon & Poli, 1997) is that a solution can be represented by numerous counter parts which are more in number, and this tends to increase tree size having same fitness but more complexities (Soule & Heckendorn, 2002) states that existence of introns (inefficient code) in trees acts as a mask and subtrees of any size can be inserted or deleted from within this part with no effect on fitness. This gives larger trees with introns, selective advantage over smaller trees, having no introns. Another theory (Tackett, 1994) states that introns act as hitchhikers and become attached to good building blocks and become a part of subtrees exchanged during crossover. Luke (2000) observed that the depth of a subtree is correlated to fitness. If a node is altered in a tree, deeper node will have less effect on the fitness of constituent tree. Therefore larger trees benefit from selective advantage and become excessive in the population.

In literature we find many instances where researchers have proposed various approaches and techniques to resolve the

Please cite this article in press as: Jabeen, H., & Baig, A. R. DepthLimited crossover in GP for classifier evolution. *Computers in Human Behavior* (2010), doi:10.1016/j.chb.2010.10.011

problem of code bloat. One of the "simplest and most frequently used (Poli et al., 2008)" method is parsimony pressure. This method penalizes the size of trees during evolution; the fitness function is a combination of actual fitness and its size. This pressure is usually kept fixed during the evolution but variable pressures have also been proposed (Langdon, 2000). The major drawback of parametric methods like this is to find exact value of the parameter.

Other methods include changes in operators to avoid bloat during evolution. We will discuss some variants of GP crossover operator proposed to avoid bloat or increase fitness during evolution.

2.3. Crossover variants in GP

Langdon (2000) presented an efficient method to avoid code bloat "size fair crossover". A subtree is selected randomly from one parent; the other subtree is selected based upon a bound placed on amount of genetic change in one operation. The subtrees bigger than $1 + 2^*$ (subtree to be deleted) cannot be replaced by first subtree. This method has proved effective in term of code bloat and efficiency. But one has to make an exhaustive search of all the subtrees present in the second parent in order to fulfill this bound. In homologous crossover (Langdon, 2000), the method is same except it deterministically chooses subtrees that are similar in position to be replaced. Both methods were found efficient in decreasing bloat without any detrimental effect on fitness.

Context aware crossover (Majeed & Ryan, 2007) attempts to preserve context of the subtrees being swapped. It selects the subtrees by comparing and selecting most similar subtrees for crossover operation.

A fitness conscious crossover is presented by Tackett named brood recombination crossover (Tackett, 1994). A brood '*n*' is created and the process of crossover between two parents is repeated '*n*' times, i.e. 2*n* children are created and the best two are selected to be injected into new population and all others are discarded. This increases the evaluation of Genetic programs '*n*' times. This problem is tackled by using only a subset of data to evaluate fitness. This attempts to eliminate the destructive effects of crossover but makes no effort to reduce tree size and the only emphasis is fitness of resulting trees.

A technique named depth dependent crossover was proposed in (Ito, Iba, & Sato, 1998). A crossover point is selected based upon depth selection probability. This is probability of selecting a certain depth for crossover. And it is higher for a node closer to root; i.e. selection of larger subtrees is favored. This method "protects building blocks and constructs larger building blocks easily by swapping higher nodes frequently" (Ito et al., 1998). This method uses a user defined parameter 'depth probabilities', in later work, this probability is adjusted using a self tuning mechanism. Both methods make no effort to reduce tree complexities during evolution, as they try to secure larger building blocks during evolution.

Next section presents the proposed crossover technique to avoid bloat for classifier evolution.

3. DepthLimited crossover

In this paper we have proposed and shown the significance of DepthLimited crossover for the problem of data classification. This method avoids exponential increase in population size during evolution, without compromising the performance of subsequent trees. In the classification method used to evolve GP (Kishore et al., 2000), one classifier is evolved for each class and output of all classifiers is combined to get the final classification decision. In case of binary classification only one classifier is evolved and in case of more than two class classification, classifiers equal to the number of classes in the data are evolved.

3.1. Initialization

Initialization plays an important role in success of any evolutionary algorithm. A diverse and efficient initialization technique can lead to effective search during the evolutionary process. We have used the well known *Ramped half and half* method (Koza, 1992) for initialization of the population. The ramped half and half method utilizes advantages of both full and grow initialization schemes with equal probability for each depth level till the maximum allowed depth. This method has been widely used for initialization in classification problems (Kishore et al., 2000).

3.2. Initial maximum depth

The initial maximum depth plays an important role in functionality of DepthLimited crossover. If the initial limits are not defined wisely the algorithm may fail to attain genetic diversity and converge. This problem becomes more important in the case of data classification where each dataset has different characteristics. In such a scenario it cannot be assumed that a single depth value will be perfect for all the datasets. The classifier represents relationships between attributes of data, it would be intuitive to define a maximum depth that should include all the attributes present in

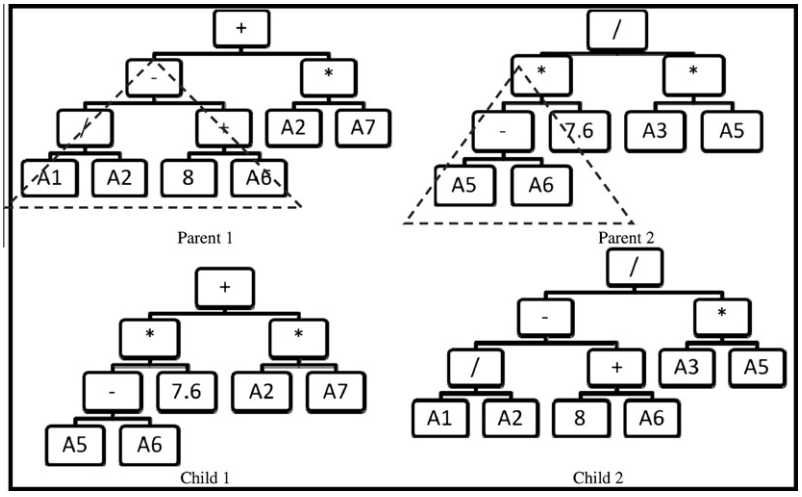

Fig. 1. DepthLimited crossover operator.

Please cite this article in press as: Jabeen, H., & Baig, A. R. DepthLimited crossover in GP for classifier evolution. *Computers in Human Behavior* (2010), doi:10.1016/j.chb.2010.10.011

the data. Let A_d be the number of attributes present in the data then such initial depth can be

$$\text{depth}_p = \text{ceil}(\log_2(A_d)) \quad (1)$$

where depth_p is the maximum depth for initialization and ceil is a function that rounds up the value to the highest integer. A full tree of depth_p depth can contain all the attributes of the data. For example, consider a dataset with 23 attributes.

$$\text{depth}_p = \text{ceil}(\log_2(23)) = 5 \quad (2)$$

A full tree with depth five has 32 leaf nodes. And such a tree can contain all the twenty-three attributes present in the data under consideration. As already mentioned, GP has the flexibility to model the underlying data, it can effectively eliminate the unnecessary attributes performing the task of feature selection. On the other hand one attribute can be used in more than one terminal node if it is adds to the fitness or it is more important. Therefore depth_p has been used as maximum allowed depth for ramped half and half initialization in our approach.

3.3. Operators and selection

Three operators, mutation, reproduction and crossover have been used for evolution. The mutation operator used in our approach is point mutation where a random node is selected and replaced by a random counterpart. A function node is replaced by a random function node and a terminal node is replaced by a random terminal node (Muni et al., 2004). The GP tree for mutation is selected randomly. The reproduction operator selects a tree based on proportionate selection and copies that tree into next generation. During the evolution, once the crossover probability is met the proposed DepthLimited crossover is performed. Two parents are selected using tournament selection. The first subtree is randomly selected from parent with smaller depth. This is to ensure that the second subtree with same depth exists in the other parent. The only restriction placed on the selection of second subtree is that it should have same depth as the first subtree. The algorithm is explained in *algorithm DepthLimited crossover* and illustrated in Fig. 1.

Algorithm DepthLimited crossover

Step 1. Begin
Step 2. Select two parents P_1, P_2 through selection mechanism
Step 3. Calculate depths D_1, D_2 of both parents.
Step 4. If ($D_1 > D_2$)
  a. Choose a random subtree S_2 from P_2
  b. Calculate depth of subtree DS_2
  c. Choose a random subtree S_1 from P_1 such that DS_1=DS_2
  d. Swap S_1 and S_2 to create two children C_1 and C_2
Step 5. Else
  a. Choose a random subtree S_1 from P_1
  b. Calculate depth of subtree DS_1
  c. Choose a random subtree S_2 from P_2 such that DS_2=DS_1
  d. Swap S_1 and S_2 to create two children C_1 and C_2
Step 6. End if
Step 7. Return C_1 and C_2
Step 8. End

3.4. Fitness

The classifier is trained to output a positive response (accept) for the class under consideration and negative response (reject) for the instances belonging to the other class. The fitness measure used during the evolution is classification accuracy. The two layered fitness is used (Freitas, 1997; Muni et al., 2004). The classifier with better accuracy is always preferred and if the accuracy of two classifiers is equal, then, the one with fewer numbers of nodes is selected. The fitness algorithm is explained in *algorithm fitness*.

Algorithm fitness

Step 1. Begin
Step 2. int countcorrect=0
Step 3. For all instances in the training data N
  a. Evaluate the classifier expression using the attribute values from the given instance (Value)
  b. If Value=0 and class=desired class
    i. countcorrect
  c. if Value0 and class=not desired class
    ii. count correct
  d. End if
Step 4. End for
Step 5. Fitness=(countcorrect/N)*100
Step 6. End

3.5. Classifier evolution algorithm

The detailed description of the classifier evolution algorithm is given in *algorithm classification*. This method has been proved efficient for data classification and has been used by several researchers with little modifications (Muni et al., 2004; Smart & Zhang, 2005). The output of this phase is an expression that is trained to differentiate between two classes by its response. It would output zero or greater value for one class and negative value for the other class.

Algorithm classification

Step 1. Begin
Step 2. For classes C=0 to N
  a. Consider C and C' as a binary classification problem, Where C'=[0 to N]-C
  b. Initialize trees using ramped half and half method using allowed maximum depth.
  c. Repeat (till maximum generations or maximum fitness achieved)
    i. Assign fitness to each tree.
    ii. Perform evolutionary operators.
    iii. Update best result
  d. Output best classifier found so far
Step 3. End for
Step 4. Combine classifiers for all classes
Step 5. Check accuracy of combined classifiers using test data
Step 6. End

The DepthLimited crossover do not let search space expand beyond predefined limits, and larger solutions do not get chance to overwhelm the population. Therefore even if multiple solutions with same fitness exist, they do not get a chance to monopolize the population. Larger trees can get selective advantage only if they are fitter, but they will not be able to produce even larger trees. The depth of children remain same as their parents.

This process of exchanging subtrees of same depth resembles the traditional crossover operator in Genetic Algorithms where

Please cite this article in press as: Jabeen, H., & Baig, A. R. DepthLimited crossover in GP for classifier evolution. *Computers in Human Behavior* (2010), doi:10.1016/j.chb.2010.10.011

fixed sized chunks are usually swapped to keep the structure of the chromosome valid.

4. Results

We have used seventeen benchmark classification problems. Sixteen problems are taken from UCI ML repository (Asuncion & Newman, 2007). One of the dataset named Ripley's data has been taken from another source (Ripley).

The datasets have been chosen from various dimensions of life to show the applicability of GP classification as well as robustness of proposed technique. All the data used for classification is real/integer valued except the BUPA dataset which has a categorical attribute with numerical values. This attribute is treated as a numerical attribute as done in Muni et al. (2004). Similarly Statlog (HRT) dataset has few binary and nominal values which were treated as numeric values. In case of WBC data, there are some missing values, which have been deleted. The datasets and their properties are summarized in Table 1.

All the reported results have been averaged for 100 executions where tenfold cross validation is applied on ten different partitioning of data twice.

The experimental parameters used for GP algorithm are presented in the Table 2. Depth_p value for DepthLimited crossover has been selected empirically as presented in Table 3.

Table 1
Datasets used for experiments.

Dataset	Attributes	Instances
WBC	10	699
BUPA	7	345
HABER	3	306
PARK	23	197
PIMA	8	786
TRANS	5	748
ION	34	351
SPEC	44	267
RIPR	3	1250
SONAR	60	208
MUSK	168	476
HRT	13	270
IRIS	3	150
WINE	12	178
VEHICLE	18	946
GLASS	10	214
YEAST	8	1484

Table 2
GP parameters for data classification.

GP parameters	
Population size	600
Crossover rate	0.50
Mutation rate	0.25
Reproduction rate	0.25
Selection for cross over	Tournament selection with size 7
Selection for mutation	Random
Selection for reproduction	Fitness proportionate selection
Mutation type	Point mutation
Initialization method	Ramped half and half method
Maximum depth	Depth_p for DepthLimited Eq. (1) and 6 for GP with no depth limits
Function set	+, –, *, and/(protected division)
Terminals	Data attributes $A_1, A_2, \ldots, A_n$, ephemeral constant [0,10]
Termination criteria	User specified generations or 100% training accuracy of classifier

Table 3
Accuracies achieved using different initial depth limits in DepthLimited GP.

Max-depth	BUPA (%)	HABER (%)	ION (%)	MUSK (%)	HRT (%)
Depth_{p-1}	61.0	68.0	73.6	53.5	72.9
Depth_p	63.6	73.6	88.5	69.3	73.3
Depth_{p+1}	59.0	72.5	75.9	63.8	73.1

Initially defined depth plays an important role in searching for desired solution, because we do not allow GP to search beyond its initial limits.

Experiments are performed using five datasets having different number of attributes and instances. As presented in Table 3, the depth_p initial depth has performed better making it a feasible initial maximum depth value.

We have used three measures to show effectiveness of the proposed approach

- Classification accuracy
- Code complexity
- Classifier simplicity

4.1. Classification accuracy

Table 4 summarizes the testing and training accuracy obtained after evolution of 120 GP generations with no bounds on tree sizes using standard GP. It is clearly visible that DepthLimited crossover has yielded compatible results as compared to the standard GP and other GP based classification methods. This can also be noted that number of average nodes of resulting classifiers is much smaller in case of DepthLimited crossover. However larger initial maximum

Table 4
Comparison of DepthLimited GP with no depth limits.

Dataset	GP without depth limitation			DepthLimited crossover GP		
	Train (%)	Test (%)	Nodes	Train (%)	Test (%)	Nodes
WBC	92.8	94.5	958	97.7	96.6	31
BUPA	69.3	68.0	3300	74.6	69.2	9.3
HABER	73.4	71.4	1075	78.1	72.5	6.5
PARK	85.6	82.6	3000	86.6	84.3	31.6
PIMA	67.8	66.4	1515	72.2	68.6	10.8
TRANS	74.3	73.8	4000	78.4	77.4	10.9
ION	86.3	85.4	1154	92.4	88.5	109
SPEC	76.0	83.4	5118	81.9	77.6	108
RIPER	88.6	88.3	946.5	89.8	89.1	6.5
SONAR	72.3	69.6	2303	79.0	73.3	110
MUSK	68.8	68.0	1482	74.3	69.3	127
HRT	79.1	72.3	1800	84.1	88.3	30.8
IRIS	99.2	93.50	1023	98.1	96.00	11.3
WINE	86.5	75.90	1987	82.2	78.50	10.2
VEHICLE	56.4	46.00	1765	68.4	49.10	18.3
GLASS	55.3	52.30	3897	64.9	54.70	13.6
YEAST	54.6	48.00	4532	65.5	57.00	20.4

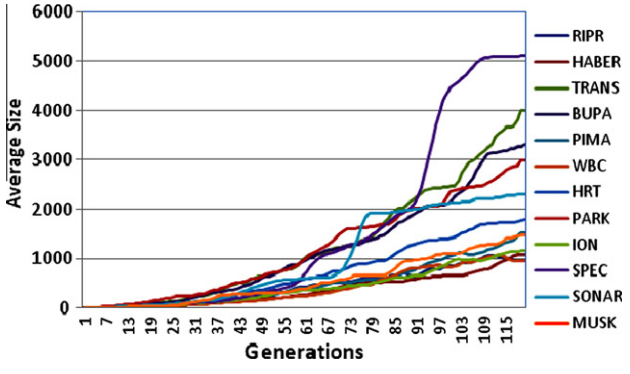

Fig. 2. Increase in population complexity during evolution for GP with no limits.

Please cite this article in press as: Jabeen, H., & Baig, A. R. DepthLimited crossover in GP for classifier evolution. *Computers in Human Behavior* (2010), doi:10.1016/j.chb.2010.10.011

depth may result in larger number of average nodes in initial generations.

4.2. Code complexities

Fig. 2 and 3 show the average population size during GP evolution using standard GP with no limits and DepthLimited crossover. Comparison of number of average nodes discloses a noticeable difference in average number of nodes. In case of traditional GP where no restrictions have been placed on evolving population size the complexity increases unlimitedly. The number of nodes per tree has reached up to 5000 in the worst case. On the other hand, for DepthLimited GP the average number of nodes cannot increase beyond the nodes present in initial maximum depth limit. The highest number of nodes can be noted in case of MUSK which has highest number of attributes and larger maximum depth. This shows that simple code can be evolved using DepthLimited crossover in GP.

4.3. Simplicity of classifiers

Fig. 4 compares the average number of nodes present in the evolved classifiers. It is clear that the DepthLimited GP method yields much simpler classifiers with compatible classification accuracies. Simpler expressions are desirable in the case of classification because they have the ability to generalize the training data instead of over fitting the data. Complex equations are usually considered to over-fit the training data and possess poor generalization capability (Kotsiantis, 2007). Moreover simpler equations require less computational power to generate the classification result. In case of standard GP, the maximum number of nodes in classifier trees reached up to 4500, whereas it never exceeded 150 nodes per tree for the proposed DepthLimited GP method; indicating improvement in terms of simplicity.

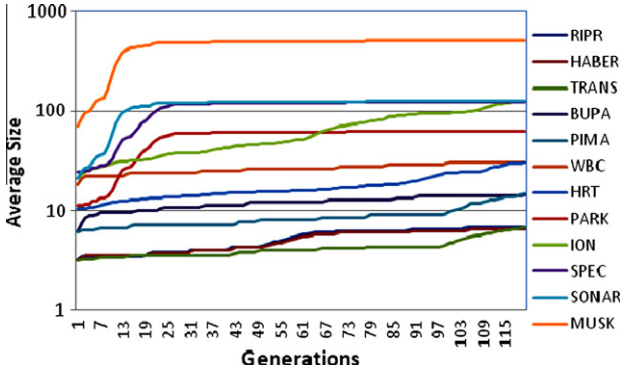

Fig. 3. Increase in population complexity during evolution for DepthLimited GP.

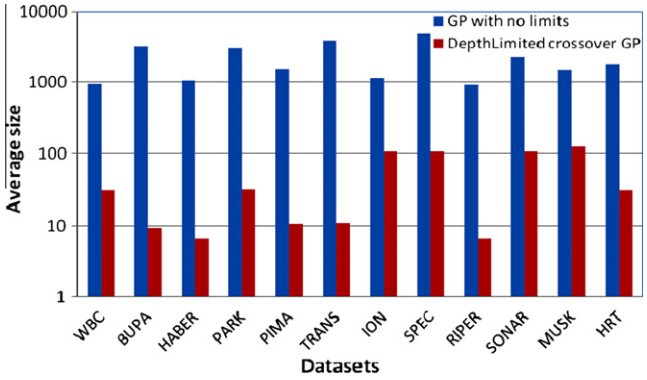

Fig. 4. Average number of nodes in classifiers.

Table 5
Comparison of DepthLimited GP with other GP based approaches.

Datasets	DepthLimited crossover GP (%)	Others
WBC	96.6	95.1% (Muni et al., 2004), 96.4% (Loveard & Ciesielski, 2001), 93.92–97.54%, (Falco, Cioppa, & Tarantino, 2002), 97.9% (Winkler, Affenzeller, & Wagner, 2009), 98.2% (Zhang & Wong, 2008), 92.5–96.24% (Tsakonas, 2006)
BUPA	69.2	68.4% (Loveard & Ciesielski, 2001), 60.8% (Muni et al., 2004)
PIMA	68.6	74.2% (Loveard & Ciesielski, 2001), 68.3–75.75% (Tsakonas, 2006), 68.64–75.16% (Falco et al., 2002), 74.87–75.53% (Winkler et al., 2009)
ION	88.5	85.4–90.52% (Pappa & Freitas, 2008)
SPEC	77.6	83.2 ± 7.3 (Zhang & Wong, 2008)
SONAR	73.3	68.4–72.42% (Pappa & Freitas, 2008)
HRT	88.3	71.4–79.66% (Tsakonas, 2006), 78.01–92.81% (Falco et al., 2002), 74.2–77.9% (Pappa & Freitas, 2008)
VEHICLE	49.1	48.6% (Muni et al., 2004), 62.2% (Loveard & Ciesielski, 2001)
IRIS	96.0	96.6% (Muni et al., 2004)

4.4. Comparison with other methods

In this section we compare the presented technique with other GP based techniques present in the literature. GP requires very long training times making it difficult to implement and experiment all the techniques present in the literature. To overcome this limitation we have implemented the proposed technique using tenfold cross validation on ten different random shuffling of the data increasing the rigorousness of proposed algorithm and keeping it consistent with the experimentations performed in other techniques. This has enabled us to compare our method with other reported results as presented in the literature.

Table 5 shows that proposed DepthLimited crossover has achieved compatible performance when compared to similar techniques.

5. Conclusion

The DepthLimited crossover has proved efficient in term of lesser computation, simpler classifiers and compatible performance. Yet it has a limitation of not being able to search beyond initially defined limits. Therefore initial limits must be selected in a way that GP can efficiently search in this predefined space with good results. This phenomenon is obvious from the presented results.

Future works include testing the proposed technique over other application areas of GP. A flexible method that can allow population to extend beyond initial limits, by automatic detection of loss of genetic diversity, can be used to overcome the limitations of proposed method.

Acknowledgement

The authors Hajira Jabeen, 041-104032-Cu-038 and Dr. Rauf Baig would like to acknowledge Higher Education Commission (HEC), Pakistan, for providing funds and resources to complete this work.

References

Asuncion, A., & Newman, D. J. (2007). Machine learning repository. Retrieved from <http://archive.ics.uci.edu/ml/>.

Carreno, E., Leguizamon, G., & Wagner, N. (2007). Evolution of classification rules for comprehensible knowledge discovery. IEEE Congress on Evolutionary Computation.

Eggermont, J. (2005). Data mining using genetic programming: Classification and symbolic regression. Ph.D. thesis, Institute for Programming Research and Algorithmics, Leiden Institute of Advanced Computer Science, Faculty of Mathematics & Natural Sciences, Leiden University, The Netherlands.

Engelbrecht, Rouwhorst, Scheoman (2002). A building block approach to genetic programming for rule discovery. In: Abbass, Hussein A., Sarker, Ruhul A., & Newton, Charles Sinclair (Eds.), Datamining.

Falco, I. D., Cioppa, A. D., & Tarantino, E. (2002). Discovering interesting classification rules with genetic programming. *Applied Soft Computing, 257*, 269.

Freitas, A. A. (1997). A genetic programming framework for two data mining tasks: Classification and generalized rule induction. Genetic Programming.

Ito, T., Iba, H., Sato, S. (1998). Depth-dependent crossover for genetic programming. IEEE Conference on Evolutionary Computation.

Johansson, U., Niklasson, L., König, R. (2007). Genetic programming – A tool for flexible rule extraction. IEEE Congress on Evolutionary Computation.

Kishore, J. K., Patnaik, L. M., Mani, V., & Agrawal, V. K. (2000). Application of genetic programming for multicategory pattern classification. *IEEE Transactions on Evolutionary Computation, 4*(3), 242–258.

Kotsiantis, S. B. (2007). Supervised machine learning: A review of classification techniques. Informatica, 31.

Koza, J. R. (1992). Genetic programming: On the programming of computers by means of natural selection. MIT Press.

Langdon, W. B. (2000). Size fair and homologous tree genetic programming crossovers. *Genetic Programming and Evolvable Machines, 1*(1), 95–119.

Langdon, W. B., & Poli, R. (1997). *Fitness causes bloat, technical report Csrp. 97–09*. Bermingham, UK: University of Bermingham.

Loveard, T., & Ciesielski, V. (2001). Representing classification problems in genetic programming. IEEE Congress on Evolutionary Computation.

Luke, S. (2000). Issues in scaling genetic programming: Breeding strategies, tree generation, and code bloat. Ph.D. thesis, Department of Computer Science, University of Maryland.

Majeed, H., & Ryanm, C. (2007). On the constructiveness of context-aware crossover. In: Proceedings of the 9th annual conference on genetic and evolutionary computation.

Muni, D. P., Pal, N. R., & Das, J. (2004). A novel approach to design classifiers using genetic programming. IEEE Transactions on Evolutionary Computation, 8(2).

Pappa, G. L., & Freitas, A. A. (2006). Automatically evolving rule induction algorithms, 17th european conference on machine learning.

Pappa, G. A., & Freitas, A. A. (2008). Evolving rule induction algorithms with multiobjective grammer based genetic programming. *Knowledge and Information Systems*.

Poli, R., Langdon, W. B., & McPhee, N. F. (2008). Field guide to genetic programming (1st ed.).

Ripley, B. D. (n.d.). Retrieved from <http://www.stats.ox.ac.uk/pub/PRNN/>.

Smart, W., & Zhang, M. (2005). Using genetic programming for multiclass classification by simultaneously solving component binary classification problems, 8th European conference on genetic programming (EuroGP).

Soule, T., & Heckendorn, R. B. (2002). An analysis of the causes of code growth in genetic programming. *Genetic Programming and Evolvable Machines, 3*(1), 283–309.

Tackett, W. A. (1994). Recombination, selection and the genetic construction of computer programs. Ph.D. thesis, University of Southern California, Department of Electrical Engineering Systems.

Tsakonas, A. (2006). A comparison of classification accuracy of four genetic programming-evolved intelligent structures. *Information Sciences, 691*, 724.

Winkler, S. M., Affenzeller, M., & Wagner, S. (2009). Using enhanced genetic programming techniques for evolving classifiers in the context of medical diagnosis. *Genetic Programming and Evolvable Machines, 111*, 140.

Yang, M., & Smart, W. (2004). *Multiclass object classification using genetic programming, CS-TR-04-2*. New Zealand: Victoria University of Wellington.

Zhang, M., & Wong, P. (2008). Genetic programming for medical classification: A program simplification approach. *Genetic Programming and Evolvable Machines, 9*(3), 229–255.

Please cite this article in press as: Jabeen, H., & Baig, A. R. DepthLimited crossover in GP for classifier evolution. *Computers in Human Behavior* (2010), doi:10.1016/j.chb.2010.10.011