Review of Classification Using Genetic Programming

HAJIRA JABEEN* AND ABDUL RAUF BAIG
National University of Computer and Emerging Sciences, Islamabad, Pakistan

ABSTRACT

Genetic programming (GP) is a powerful evolutionary algorithm introduced to evolve computer programs automatically. It is a domain independent, stochastic method with an important ability to represent programs of arbitrary size and shape. Its flexible nature has attracted numerous researchers in data mining community to use GP for classification. In this paper we have reviewed and analyzed tree based GP classification methods and propose taxonomy of these methods. We have also discussed various strengths and weaknesses of the technique and provide a framework to optimize the task of GP based classification.

Keywords: Data Classification, Genetic Programming, Survey, Taxonomy

1. INTRODUCTION

Genetic Programming (GP) introduced by Koza in 1992 is an evolutionary algorithm designed for automatically constructing and evolving computer programs. This innovative flexible and interesting technique has been applied to solve numerous interesting problems. Classification is one of the ways to model the problems of face recognition, speech recognition, fraud detection and knowledge extraction from databases.

Data Classification can be defined as assigning a class label to a data instance based upon knowledge gained from previously seen class labeled data. Various classification algorithms have been proposed and are being used depending upon their simplicity, understandability or accuracy. Simpler techniques like decision trees are simple and understandable but applicable to small data sets only. On the other hand statistical techniques or Neural Networks are not easily comprehensible.

Evolutionary algorithms like Genetic Algorithms (GA) (1) have been found successful in solving classification problems. GP has emerged as an extension of GA proposed by Cramer (2) and Schmidhuber (3). GP differ from GA in the ability to evolve variable length solutions (computer programs). Later, Koza (4) used the term GP and popularized this technique as a new evolutionary algorithm rather than an extension of GA. GP has emerged as a powerful tool for classifier evolution. To date, many variations of GP have been introduced to handle the classification, this includes Linear GP, Grammar based GP, Graph based GP and Tree based GP (5). These variations differ in representations of solutions.

GP works by evolving a population of randomly created initial programs using a fitness measure. It selects fitter ones to take part in the evolution to efficiently search for desired efficient solution. The basic GP algorithm is similar to any evolutionary algorithms and works as follows.

Algorithm GP Evolution

```
Step 1. Begin
Step 2. Define pop-size as desired population size
Step 3. Randomly initialize pop-size population
Step 4. While (Ideal best found or certain number of generations met)
  o     Evaluate fitness
  o     While(number of children=population size)
  o     Select parents
  o     Apply evolutionary operators to create children
  o     End while
Step 5. End While
Step 6. Return Best solution
Step 7. End
```

The following are the main steps involved to use GP for solving any problem.

1.1 Problem Representation

The representation of an individual is the method to construct the solution for a desired problem. This can also be termed as the data structure used to define an individual. The representations used in GP can be divided into following types.

Trees Based GP

It is the most common representation used in GP. Trees can also be represented as LISP statements in which data and code are closely related although prefix notations or pointer based representations can also be used in some languages. In such cases, each individual (phenotype) must be executed using the data that constitutes the genotype of the individual. In such case all the data pairs are executed against the individual and the return values are used to calculate the corresponding accuracy or error, representing the fitness of the tree.

Constrained Syntax GP

Instead of simple binary trees, the trees might be needed where complex functions (like "if" having more than two arguments) are required. In such trees, some constraints must be placed on the genetic operators to maintain the validity of the tree after the operator has been performed upon.

Cellular GP

In cellular or indirect encoding, the trees represent programs that direct the creation of the second structure which is usually a graph structure, like neural networks or petrinets. A slightly modified form named edge encoding is also used to represent planar and simple graph structures.

Linear GP

Another important type of GP representations is the list of machine language instructions. Linear GP and Grammatical evolution in GP use this type of representations.

Graph based GP

It is one of the most complex representation structures. These are usually used to represent and evolve neural networks, automata or petrinets.

Grammar based GP

This is another type of representation where a set of production rules are defined to use in creating population members.

The focus of this thesis is the simple tree based representations that are created using a predefined set of terminals and functions in a primitive set. Such a representation is simple and has been used frequently for the data classification problem.

1.2 Solution Initialization

The innovation of GP lies in the variable sized solution representation which requires efficient initial population construction, this feature makes it different from other evolutionary algorithms. Individuals are represented as trees constructed randomly from a primitive set. This primitive set contains functions and terminals. A tree's internal nodes are selected from the functions and leaf nodes are selected from the terminals. GP allows variety in composition of solution structures using same primitive set.

Initialization plays an important role in success of an evolutionary algorithm. A poor initial population can cause any good algorithm to get stuck in local optima. On the other hand a good initialization can make most of the algorithms work sufficiently well. There are few initialization techniques popular in tree based GP.

Full Method

This method enforces construction of full trees up to the defined depth. The tree is created by selecting function nodes only till the allowed depth. After this depth the nodes are selected from the terminal set only. This method forces all trees to be full.

Grow Method

Grow method randomly selects nodes from function or terminal set and creates random trees till maximum depth-1 achieved, after that only terminal nodes are selected to keep the tree-depth fixed. The trees created with such method vary in their structure due to freedom in selection.

Ramped Half and Half Method

Koza (4) proposed a combination of full and grow methods to overcome the disadvantages of both methods. The

ramped half and half method ramps the number of trees to be created, to the maximum depth and for each depth, trees are randomly created using either the full or grow method. This initialization scheme produces diverse and bushier trees. The method has been widely applied and found successful.
Some other methods for tree initialization are ramped uniform initialization (6) and PTC2 initialization (7).

1.3 Selection

The evolutionary operators are applied on individuals particularly selected for that operation. The individuals are selected using a particular selection mechanism. Two of such mechanisms are defined as follows.

Tournament Selection

In this type of selection, a tournament is conducted among few individuals chosen randomly from the population. The winner or best member is selected as a result of a tournament. The tournament size determines how many random members are selected for the tournament. Tournament size determines the selective pressure; large tournament size favors fitter solutions for selection.

Fitness Proportionate Selection

All the trees have probability of selection based upon their fitness. The probability of selection for a population of size 'N' is calculated as

$$P_i = f_i / \sum_{i=1}^{n} f_i \qquad 1$$

This is also called Roulette Wheel Selection mechanism.
Several other selection mechanisms also exist in the literature like Rank Based Selection and Stochastic Universal Sampling.

1.4 Operators

The most common operators used for evolution of GP programs are crossover, mutation and reproduction. Each of these will be discussed in the coming sections.

Crossover

Crossover operator works by selecting two parents from the population. Two random subtrees are selected from each parent and swapped to create children. Advancements have been made to pure random crossover operator in order to make it more efficient and propagate good building blocks among generations. The information regarding size (6), depth (8), location (9) or homogeneity (6) of subtrees is also exploited while performing this operation.

Mutation

Mutation used in GP is of three types
In **point mutation** a single node in parent tree is selected and replaced with a random node of same type. E.g. a function node is replaced by a function node of same arity and a terminal node is replaced by a randomly selected terminal node.
Shrink mutation selects a node randomly and the subtree rooted at that node is replaced by a single terminal node.
Grow mutation selects a random node and a randomly generated subtree is replaced by the subtree rooted at that node.

Reproduction

In this operator an individual is selected and copied directly to the new generation without any changes or modifications to it.

1.5 Solution Fitness

Fitness is the performance of an individual corresponding to the problem it is aimed to solve. It tells which elements or the regions of the search space are good. The fitness measure steers the evolutionary process towards better approximate solutions to the problem. Fitness of individuals in a population can be measured in many ways. It can be measure of error between the original and desired output of a solution. It can be compliance of the structure to the task it is required to solve based on a user specified criteria. The difference between fitness evaluation in GP and other evolutionary algorithms is that each individual of GP is a *program* which needs recursive execution of the nodes of the tree in precise manner. This adds overhead to the algorithm increasing its evolution time and required computational sources.

1.6 Termination Condition

The above mentioned steps are applied during the evolution process in a recursive manner refining the solutions from generation to generation. The termination condition determines when this iterative process needs to be stopped. The commonly used termination criteria are completion of a given number of generations or success in finding a solution of desired fitness.

GP has been considered a useful technique for classification since its inception (4). This paper aims at providing an overview of recent work, relevant to classification, and discusses the advancements made to date. We have also discussed the related issues that need to be addressed for classifier evolution. Next section provides an over view of classification methods proposed so far and classify them into three main categories. We do not claim to provide exhaustive overview of methods applied to GP based classification methods. We have excluded graph based and linear GP based classification methods in this paper. We have tried our best to cover most of the tree based and grammar based methods applied to classification task. However, we present some weakness of the task and a framework to overcome them.

2 CLASSIFICATION USING GP

Several techniques have been proposed to tackle classification using GP. We have categorized these classification methods into three types. First type describes the evolution of classification algorithms like decision trees, neural networks or other rule induction induction algorithms. This method truly portrays the use of GP for program evolution (10). The other method includes evolution of classification rules or expressions. Rules are evolved in the form of logical expressions with logical operators. In another type the expressions are evolved in the form of arithmetic expressions or functions.

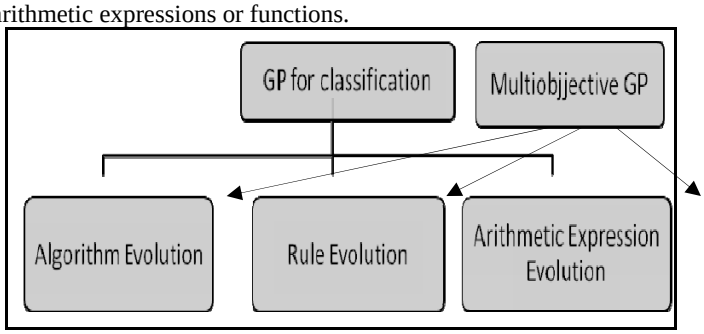

Figure 1 Taxonomy of GP for Classification

2.1 Evolution of Classification Algorithms

GP has been used to evolve classification algorithms like decision trees, fuzzy decision trees, neural networks and other rule induction algorithms. For such systems a grammar or a set of rules are predefined. Random solutions are initialized using these rules. The structures of solution are designed in a way to remain valid after application of genetic operators like crossover and mutation to efficiently search the solution space for optimal results. This involves defining some specialised and constrained crossover/mutation operators.

Decision trees are the simple classifiers and GP has been extensively used to evolve them. The work ranges from Koza's explanation (11) to recent (12). Marmelstein (13) and Bojarczuk (14) used standard GP operators to evolve decision trees using a defined syntax. Folino (15) used a hybrid GP and simulated annealing system to evolve decision trees. Bot (16) has used GP to evolve oblique decision trees, where the functions in the nodes of the trees can use one or more variables. In the buildingblock (17) approach, decision trees are built from simpler to complex trees during evolution. Eggermont (18) used 'atomic' representation to represent decision trees. A two layered fitness is used to evolve these trees to prefer smaller trees over larger with same accuracy. A parallel GP based approach has been used by Folino using the concept of cellular GP for Decision Tree evolution (19) . Decision tree evolution methods suffer from the drawbacks of decision trees. Decision trees are applicable to categorical data only. Their efficiency in disturbed if the training data is too small or too large making decision trees unstable. Moreover decision tree can become very large requiring further steps for detection and pruning of such parts.

Tsakonas (20) has evolved intelligent structures for classification. He used grammar based GP and presented a context free grammar for evolution of decision trees, fuzzy rule based system, feed forward neural networks and fuzzy petrinets. Neural Networks and Fuzzy Petrinets are expressed by applying cellular encoding. He used six datasets to show the applicability of GP evolved intelligent structures for knowledge discovery.

The Rule Induction system (21) used grammar based GP where a grammar is used to define rule induction algorithms which are automatically evolved using GP. These Algorithms have been found compatible to

manually designed rule induction algorithms such as RIPPER and C4.5. These GP evolved algorithms have been tested on real world problems and have achieved comparable performance.

Autonomous GP Solver (22) has been proposed recently that can construct solutions, store and update existing solutions by using an adaptive variant of GP. This autonomous system is able to decide if it knows to solve the problem or not. The proposed system is able to handle classification and regression problems.

Although the evolution of classification structures is an innovative idea, yet the evolved neural networks, decision trees or other algorithms do not overcome the basic disadvantages suffered by evolved algorithms like neural networks. On the other hand, another layer of difficulty is added to such structures. The above mentioned techniques are dependent upon the flexibility and expression of underlying grammar and the operators are somewhat constrained to keep the structures of the solutions valid. Grammar based method helps in avoiding evolution of meaningless solutions, and reduces the search space among valid candidates only. But this might compromise the flexible nature of GP evolution. An inefficient grammar might introduce more constraints biasing the efficiency of search process.

2.2 Evolution of Classification Rules

In this section we will discuss some state of the art strategies used to extract logical classification rules. These are usually in the form of If-Then statements. Decision trees, mentioned in the previous section, can be translated into the set of rules by creating a separate rule for each path in the tree (23). However, Individual rules can also be learned from training data. GP has been used for evolution of classification rules since long (24). In such systems an individual tree represents a single rule which is created using some predefined logical functions and terminals, where terminals define the operands of the rule (attributes values of the data) and consequence of the rule is the resultant class.

Alex Frietas (25) introduced a classic framework to use GP for data mining in 1997. A GP individual encodes SQL queries following a grammatical representation of relational database system and is named as Tuple Set Descriptor (TSD). The fitness of an individual is computed by executing these SQL queries. The advantage for SQL like representation is scalability, data privacy, no redundancy, parallel execution on SQL servers and portability across multiple domains.

Eggermont (26) introduced interesting and understandable ‘atomic’ representations for GP based classifiers. An atom is a predicate of the form (attribute operator value) where operator is a boolean function, this is also known as booleanization of data applicable to data with different types of attributes. A tree is traversed from root to leaf node to determine the class of an instance.

Wong used GP to evolve rules (27) using inductive logic programming. He introduced the LOgic grammars based GENetic PROgramming system (LOGENPRO). The basic concept has been adapted from language compilers and makes use of context free grammars to represent and evolve various rule representations utilizing different languages.

Falco (28) used GP to evolve comprehensible simple rules by combining the parallel searching ability of genetic programming. The classifier trees are constructed using logical functions and attribute values. A grammar has been designed that can represent such rules. It has been shown that the evolved rules are comprehensible, emphasize discriminating variables and achieve compatible performance as compared to other classification algorithms on benchmark datasets.

Huang (29) developed a two stage GP S2GP for classification. The system evolves IF-THEN rules in the first stage and a discriminating function for the examples not covered by first stage. This system has outperformed several conventional classification methods like CART and C4.5 for credit classification problem. Tunsel (30), Berlanga (31) and Mendes (32) introduced evolution of fuzzy rules using GP. Chien (33) used fuzzy discrimination function for classification. In (34) and (35) Bozarczuk used a GP based approach, where set of functions applicable to different type of attributes is defined to represent the rules as disjunctive normal form. Several constraints are placed on the tree structure to express a valid rule. This type of GP is also referred as constrained syntax GP. Tsakonas (36) introduced two GP based systems for medical domains and achieved noticeable performance. Lin (37) proposed a layered GP where different layers correspond to different populations performing the task of feature extraction and classification. Some other rule-based classification methods include (38), (39) and (40).

The rule evolution algorithms usually require the data to be of categorical type. If the attributes of data are of more than one type and different functions are applicable on different attributes of data, then, some constraints on tree structure are required to confirm the closure property. This is called constrained syntax GP. The other method is descritization or booleanization of data.

2.3 Evolution of Classifier Expressions

GP has gained attention for evolution of classifier expressions for numerical or real valued data. It has become popular due to its simplicity, applicability and outstanding performance. These expressions use the attributes of data as variables and serve as a discriminating function between classes. The output of such

expressions is a single real value. A threshold of positive and negative numbers can serve as a natural boundary for two class classification problems. In case of more than one class, several methods have been used, one of these is assigning thresholds. The method includes assigning static thresholds (41) (42), dynamic thresholds (43) (42) and slotted thresholds (44). Another scheme is binary decomposition. In this technique a classifier for each class is evolved separately considering other classes of data as single 'not desired' class. All the resulting best classifiers are integrated into one final classifier. Classification decision is made based on outputs of all classifiers. The classifier with positive output or maximum output is declared winner. Binary decomposition methods have been explored in (45) (46) (47) (48) (49). A relatively different, GA inspired, method for multiclass classification has been proposed by Durga (50), an amalgamated chromosome (vector) of classifiers for all the classes is evolved in single GP run. Other effective multiclass classification methods include Mende's work (51) where two populations are evolved simultaneously, one population contains fuzzy rule sets and other population contains membership functions. Both populations are coevolved so that they can effectively adapt to each other. Loveard (46) proposed and compared five different methods for multi-class classification. These methods are binary decomposition, static range selection, dynamic range selection, class accumulation and evidence accumulation. The results revealed that dynamic range selection method is efficient for multiclass classification

3 FITNESS FUNCTION FOR CLASSIFICATION

One of the most common fitness function used for classification task is the classification accuracy. The accuracy tells the number of instances correctly classified by a classifier. Another measure to minimize can be classification error which is reciprocal of classification accuracy. Both these measures are not true measures for discriminative power of a classifier and can be disturbed by the imbalance of data. To overcome this limitation of classification accuracy some researchers have also used area under the convex hull as a fitness measure to favor more discriminative rather than accurate classifiers. To evaluate the fitness of a classifier, it is evaluated for each data instance and the result adds to or decreases from its overall fitness. Some researchers have also used a combined fitness that favors smaller trees, by combining accuracy with a size penalty. The researchers have also used more than one fitness measures for classifier evolution task. Such systems are named as Multiobjective optimization and make a separate field of research. Multiobjective optimization can be used for any of the above mentioned classification type.

4 MULTIOBJECTIVE GP

This technique cannot be classified as one of the special technique for classification. This technique can be and has been used to evolve classifiers of all the above mentioned types. The main idea behind multi objective optimization is to have more than one fitness criteria for each population member and a desire to optimize the solutions for each fitness function. The solution must be acceptable with respect to all the fitness functions simultaneously. This technique is popular in GP in order to favor simpler solutions because code bloat (increase in program sizes during evolution) is a major drawback of GP.On the other hand, in case of classification a simple and accurate classifier is desired. Therefore common objectives taken into account for the task of classification are classifier size and accuracy. Lichodzijewski (52) proposed an interesting bid based approach for co-evolution of GP classifiers. A test population and a learner population are coevolved. Test population is subset of training set and each learner has a bid and an action where bid is the program (classifier) and action is classification label. The goal of learner is to correctly classify tests. And the goal of test is to accurately distinguish between the learners. In (53) two objectives, number of nodes in a tree and misclassification error were taken into account. The method was used for the classification of nominal data. Another work for network traffic classification has been performed by Ostaszewski (54) where the objective functions are sensitivity and specificity of classifiers. The classifiers obtained yielded high performance making it applicable for network security problems.

5 STRENGTHS AND WEAKNESSES

5.1 Strengths

Evolutionary algorithms have been found efficient in finding solutions to the classification problems autonomously. GP, being one of the evolutionary algorithms enjoys all benefits offered by evolutionary algorithms and adds several more. This section discusses several advantages offered by GP for classification.

GP has inherited the stochastic search properties of evolutionary algorithms and acts as a global search mechanism that makes use of hyper plane search. This makes it less likely to get stuck in the local optimum. This is different from other methods like neural networks or gradient descent which are prone to local optimal values.

GP enjoys the benefits of variety in solution structures. This is opposed to the fixed size solutions offered by

most of the evolutionary algorithms or fixed architectures of neural networks. These programs can contain numerous functions, variables and constants usually desired in various problem solving. This usually eliminates the need of having different genotype to phenotype encodings (45). This feature helps GP to search for better solutions by giving freedom of expression to search for relationships, and importance of different attributes in data. These flexible representations help GP to automatically model the inherent data dependencies in its evolving structures and the algorithm may not require any explicit information or preprocessing regarding class or attribute dependencies (45). GP can automatically eliminate attributes unnecessary for the classification performing the task of feature extraction algorithm (45). Similarly important attributes can appear near the root whereas less important ones would appear deeper in the tree (55). GP is able to operate on chunks of data to extract meaningful rules. There is no need to use all of the training data to evolve classifiers. (45) (50). A form of incremental learning has been sucessfully used for evolving classifiers by GP. The classifiers obtained through GP are usually understandable and transparent (14) (38). They are like white boxes that clearly portray the relationships of attributes required for a particular class, as opposed to many other black box solutions like neural networks (17). GP evolves the classifiers in the form of a program. We can simply evaluate the final classifier (program) for the classification decision. This helps in easy and fast interpretation of results. These expressions or programs are easily portable in tools like spread sheets or MATLAB for future data evaluation (17). The classifier representations differ in each separate execution, so we can extract several different classifiers with same or slightly different accuracy. This is also called lack of population convergence. Although undesirable in few cases, it can search variety of solutions for a same problem.

GP has the ability to operate upon the data in its original form. No preprocessing or data transformations are usually required to apply GP for classification task. For example GPCE (45) (50) can use real valued data and categorical data has been used in another application (18) for evolution of logical rules. Yet, we might need type conversions for classification of mixed type of data (56). GP based classifier evolution is not affected by the data distribution (45). This is in contrast to the neural networks which are highly dependent on the data distribution. This autonomy enables efficient discovery of unknown knowledge from the data.

Above mentioned benefits are also reported by Poli (5), in which GP is said to perform well for the problems having properties like unknown interrelationships of variables, Finding size and shape of solution is a part of problem, Test data availability, Failure of conventional mathematical analysis, Acceptability of approximate solutions, Improvements in performance in measureable and availability of simulators to measure the performance of solutions but poor methods to obtain the solution itself. We can observe that all these properties are inherent in the problem of classification, making it feasible application domain for GP. All these factors make GP rather attractive to use for classification problems. Numerous researchers have applied GP for the said task and we can find loads of work done in the field. The missing item in all this work is a meaningful categorization and analysis of these techniques. In the next section we will present the taxonomy of several classification methods present in the literature.

5.2 Weaknesses

In the previous section we have discussed numerous advantages of using GP for classification. GP suffers from a few problems as well. Most of these problems are general and not specific to the classification problem only. These issues have received lesser attention in the classification scenario in the past. Few researchers have attempted to tackle individual problems recently but a definite solution has yet to be found.

The drawback of GP is the necessity of frequent evaluation of fitness (usually recursive) of each program in the population in each generation. If we have '*N*' population size and '*E*' generations the number of fitness evaluations would be *N*E*. We know that the datasets tend to have loads of data, and a classifier must be evaluated for each instance in data, making the evaluation of an individual the most time consuming operation of the algorithm. GP suffers from well known phenomena of bloat. The sizes of evolving structures start increasing without any corresponding increase in the accuracy of programs. This increases the training time of already computational greedy task and size of resultant classifiers. On the other hand, it is commonly believed that simpler classifiers exhibit better generalization abilities. This phenomenon also affects the comprehensibility of discovered classifiers.Although considerable work has been done to tackle the problem of bloat (57) but most of this work is not in the context of classification. GP based classifiers are either applicable to nominal or numerical data. For both types of data to work, we must perform conversion from one type to another. The need arise for a robust mechanism to classify mixed types of attribute data. Different GP runs yield different results in each execution. These results usually differ in fitness as well as structure. This common property of GP is referred as lack of convergence and may prohibit a GP system to find optimal results for every execution. GP has been successful in classification for various applications. But it lacks a proper methodology that could be applied for multiclass classification. Several methods for multiclass classification have been proposed but the technique lacks a definite solution.

6 PROPOSED ADJUSTMENTS

We have proposed some possible modifications to GP targeted for classifier evolution. Some of these techniques have been attempted by few researchers individually but the work has not been integrated as a single system, or no comparison of these methods has been performed. One could consider all of these factors for classifier evolution.

The problem of long training time can be tackled by using some efficient search strategy, one example of such search mechanisms is "pyramid search 'proposed by Loveard (58). After few generations the solutions below certain fitness are eliminated from the population in assumption that they are not playing an important role in evolution and fitness enhancement. This is one such method incorporated to reduce training time for classifier evolution. Some other intelligent search methods can be devised to avoid training time. Besides reducing the population size, we can also reduce the training samples to decrease the training time. A similar method named incremental learning has been used by Muni (50) and Kishore (45) .

Bloat is the major bottle neck of GP for classifier evolution. Several methods have been proposed in literature to avoid bloat and some of them have been used in classifier evolution. Winkler (59) used sizefair crossover (6), Muni (50)and Kishore (45) used tree size limits. Eggermont (18) used two layered fitness. depthdependent (8) crossover was used by Badran (53). But none of these methods have been designed for classifier evolution, or compared with traditional GP in case of classification. One such work can be found in (60) , where depthlimited crossover operator has been proposed to eliminate bloat and evolve simpler classifiers. But there is a need for introduction of specialized crossover that does not let classifiers increase in size unnecessarily as larger classifier compromise the generalizing ability of a classifier. Zhang (61) used an online simplification approach that simplified algebraic expressions by applying reduction formulas using a hashing method. Other methods include tree complexity penalty in fitness evaluation (parsimony pressure) (5), limits on crossover operations like size fair crossover operator used for GP based classifier evolution (59) have been used for classifier evolution and limits on maximum tree size (50).

The problem of lack of convergence can be handled by using some optimization mechanism that can increase the efficiency of evolved classifiers. A unique method (62) performs gradient search for optimization of ephemeral constants or numeric constant values present in GP trees producing better results for symbolic regression. Zhang (63) applied offline and online learning method for learning of ephemeral constants in a GP tree using gradient descent for object recognition that outperformed traditional GP. The gradient descent algorithm is augmented to the existing GP system, by application on each program in population in a generation. The remaining evolutionary process remains conventional. The results conclude that online scheme offers better performance. The online learning is similar to incremental learning in Neural Networks and offline is similar to batch training in Neural Network. In another motivating work, (64) weights are added to all the edges present in a GP expression tree. These weights are then updated using gradient descent based local learning mechanism during the evolutionary process. The local learning phase is augmented with the traditional evolution of GP expressions. The method was found efficient in terms of accuracy. Both methods proposed by Zhang are coupled into the GP, increasing the complexity of already computational extensive task, although considerable increase in performance has been achieved.

The robustness problem has been handled by Loveard (56) by exploring four different techniques for using categorical attributes. These were mapping to integer values, using indicator variables, multi branching based on attribute values and if-then nodes. Later Badran (53) investigated the former two techniques and concluded that for ordered attributes integer mapping works best and for nominal attributes indicator variables yield best performance.

Finally, for the problem of more than one class, one of the proposed methods is to assign thresholds. The thresholds could be static (41) (42), dynamic (43) (42) and slotted (44). The problem with this method is that it is applicable to the expressions that output real value rather than boolean value. Another scheme for multiclass classification is binary decomposition; a classifier for each class is evolved separately considering other classes of data as single 'not desired' class. All the resulting best classifiers are integrated into one final classifier. Classification decision is made based on outputs of all classifiers. The classifier with positive output or maximum output is declared winner. Binary decomposition methods have been explored in (45) (46) (47) (48) (49) (50). The major drawback of both the approaches is the conflict between more than one classifier that should be handled intelligently.

It is suggested that all of these measures should be taken into account to come up with an efficient and robust classification method.

7 CONCLUSIONS

We have seen that GP can perform the task of classifier evolution effectively. It has achieves compatible or better performance in many instances. Besides this success GP based classifier evolution suffers from several problems like long training time, bloat and lack of convergence. The need arise for efficient optimization steps for the task of classifier evolution using GP. The present classification techniques lack robustness, any measures to decrease the training time, making the classifiers bloat free and any mechanism to overcome the problem of lack of convergence.

Although being an interesting technique applicable for data classification, GP need more attention to mature. There are only few researchers actually progressing towards GP based intelligent and autonomous classification.

8 REFERENCES

[1] *Using Genetic Algorithms for Concept Learning.* **Jong, De, Spears, W M and Gordon, D F.** 1993, Machine Learning, pp. 161-188.

[2] *A representation for the adaptive generation of simple sequential programs.* **Cramer, N L.** 1985. Proceedings of International Conference on Genetic Algorithms and the Applications.

[3] *Probabilistic Incremental Program Evolution.* **Salustowicz, R P and Schmidhuber, J.** 1987. Evolutionary Computation. pp. 123–141.

[4] *Genetic Programming: On the Programming of by Means of Natural Selection.* **Koza, J R.** Cambridge : MIT Press, 1992. MA.

[5] **Poli, R, Langdon, W B and McPhee, N F.** *A Field Guide to Genetic Programming.* 2008.

[6] *Size Fair and Homologous Tree Crossovers for Tree Genetic Programming.* **Langdon, W B.** 2000, Genetic Programming and Evolvable Machines, pp. 95-119.

[7] **Luke, S.** *Essentials of Metaheuristics.* 2009.

[8] *Depth-dependent crossover for genetic programming.* **Ito, T, Iba, H and Sato, S.** Alaska : IEEE Press, 1998. Proceedings of the 1998 IEEE World Congress on Computational Intelligence. pp. 775-780.

[9] *Context preserving crossover in genetic programming.* **Dhaeseleer, P.** Orlando, Florida, USA : IEEE, 1994. Proceedings of the 1994 IEEE World Congress on Computational Intelligence.

[10] **Freitas, A A.** *Data Mining and Knowledge Discovery with Evolutionary Algorithms.* Berlin : Spriger-Verlag, 2002.

[11] *Concept formation and decision tree induction using the genetic programming paradigm.* **Koza, J R.** s.l. : Springer-Verlag, 1991. Parallel Problem Solving from Nature,Proceeding of first workshop,Lecture Notes in Computer Science.

[12] *Dynamic Split-Point Selection Method for Decision Tree Evolved by Gene Expression Programming.* **Li, Q, et al.** 2009. IEEE Congress on Evolutionary Computation.

[13] *Pattern Classification using a Hybrid Genetic Program – Decision Tree Approach.* **Marmelstein, R E and Lamont, G B.** s.l. : Morgan Kaufmann, 1998. Proceedings of the Third Annual Conference of Genetic Programming. pp. 223-231.

[14] *Discovering Comprehensible Classification Rules using Genetic Programming: A Case Study in a Medical Domain.* **Bojarczuk, C C, Lopes, H S and Freitas, A A.** s.l. : Morgan Kaufmann, 1999. Proceedings of the Genetic and Evolutionary Computation Conference. pp. 953-958.

[15] *Genetic Programming and Simulated Annealing: A Hybrid Method to Evolve Decision Trees.* **Folino, G, Pizzyti, C and Spezzano, G.** 2000. Proceedings of the European Conference on Genetic Programming.

[16] **Bot, M.** *Application of Genetic Programming to Induction of Linear Classification Trees.* Faculty of Exact Sciences, Vrije Universiteit. Amsterdam : s.n., 1999. Final Term Project Report.

[17] **Engelbrecht, A P, Schoeman, L and Rouwhorst, S.** A Building Block Approach to Genetic Programming for Rule Discovery, in Data Mining: A Heuristic Approach. [book auth.] H A Abbass, R Sarkar and C Newton. *Data Mining.* s.l. : Idea Group Publishing, pp. 175-189.

[18] **Eggermont, J.** *Data Mining using Genetic Programming: Classification and Symbolic Regression.* Leiden University. 2005. PhD Thesis.

[19] *Parallel genetic programming for decision tree induction.* **Folino, G, Pizzuti, C and Spezzano, G.** Dallas, TX USA : IEEE, 2001. Proceedings of the 13th International Conference on Tools with Artificial Intelligence.

[20] *A comparison of classification accuracy of four genetic programming-evolved intelligent structures.* **Tsakonas, A.** 2006, Information Sciences, pp. 691-724.

[21] *Evolving Rule Induction Algorithms with Multiobjective Grammer based Genetic Programming.* **Pappa, G A and Freitas, A A.** 2008, Knowledge and Information Systems.

[22] *An autonomous GP-based system for regression and classification problems.* **Oltean, M and Diosan, L.** s.l. : Elsevier, 2009, Applied Soft Computing, Vol. 9, pp. 49-60.

[23] **Quinlan, j r.** C4.5: programs for machine learning. San Francisco : Morgan Kaufmann.

[24] **South, M C.** *The Application of Genetic Algorithms to Rule Finding in Data Analysis.* 1994.

[25] *A Genetic Programming Framework For Two Data Mining Tasks : Classification And Generalized Rule Induction.* **Freitas, A A.** CA , USA : Morgan Kaufmann, 1997. Genetic Programming. pp. 96-101.

[26] *GP For Data Classification , Partitioning The Search Space.* **Eggermont, J, Kok, J N and Kosters, W A.** 2004. Proceedings of the 2004 Symposium on Applied Computing. pp. 1001-1005.

[27] *Learning Programs in Different Paradigms using Genetic Programming.* **Wong, M L and Leung, K S.** Berlin, Germany : Springer-Verlag, 1995. Proceedings of the Fourth Congress of the Italian Association for Artificial Intelligence.

[28] *Discovering Interesting Classification Rules With GP.* **Falco, I D, Cioppa, A D and Tarantino, E.** 2002, Applied Soft Computing, pp. 257-269.

[29] *Two-stage Genetic Programming (2SGP) for the credit scoring model.* **Huang, J J, Tzeng, G H and Ong, C S.** 2, s.l. : Elsevier, 2006, Applied Mathematics and Computation, Vol. 174, pp. 1039-1053.

[30] *On Genetic Programming of Fuzzy Rule-Based Systems for Intelligent Control.* **Tunstel, E and Jamshidi, M.** 1996, International Journal of Intelligent Automation and Soft Computing, pp. 273-284.

[31] *A Genetic-Programming-Based Approach for the Learning of Compact Fuzzy Rule-Based Classification Systems.* **Berlanga, F J, et al.** 2006, Lecture Notes on Artificial Intelligence (LNAI), pp. 182-191.

[32] *Discovering Fuzzy Classification Rules with Genetic Programming and Co-Evolution.* **Mendes, R R F, et al.** s.l. : Springer, 2001, Lecture notes in Artificial Intelligence, pp. 314-325.

[33] *Learning discriminant functions with fuzzy attributes for classification using genetic programming.* **Chien, B C, Lin, J Y and Hong, T P.** 1, s.l. : Elsevier, 2002, Expert Systems with Applications, Vol. 23, pp. 31-37.
[34] *A constrained-syntax genetic programming system for discovering classification rules: application to medical data sets.* **Bojarczuk, C C, et al.** 2004, Artificial Intelligence in Medicine, pp. 27-48.
[35] *An innovative application of a constrained-syntax genetic programming system to the problem of predicting survival of patients.* **Bojarczuk, C C, Lopes, H S and Freitas, A A.** s.l. : Springer-Verlag, 2003, Lecture Notes in Computer Science, Vol. 2610.
[36] *Evolving rule-based systems in two medical domains using genetic programming.* **Tsakonas, A, et al.** 2004, Artificial Intelligence in Medicine, pp. 195-216.
[37] *Classifier design with feature selection and feature extraction using layered genetic programming.* **Lin, J Y, et al.** 2007, Expert Systems with Applications, Vol. 34, pp. 1384-1393.
[38] *Evolution of Classification Rules for Comprehensible Knowledge Discovery.* **Carreno, E, Leguizamon, G and Wagner, N.** 2007. IEEE Congress on Evolutionary Computation. pp. 1261-1268.
[39] *A comparison of genetic programming variants for data classification.* **Eggermont, J, Eiben, A E and Hemert, J I.** 1999. Proceedings of the Eleventh Belgium Netherlands conference on Artificial Intelligence. pp. 253-254.
[40] *Genetic Programming - A Tool for Flexible Rule Extraction.* **Konig, R, Johansson, U and Niklasson, L.** 2007. IEEE Congress on Evolutionary Computation.
[41] *Genetic Programming For Multiple Class object Detection.* **Zhang, M and Ciesielski, V.** Australia : s.n., 1999. Proceedings of the 12th Australian Joint Conference on Artificial Intelligence. pp. 180–192.
[42] *Multi-objective techniques in genetic programming for evolving classifiers.* **Parrott, D, Li, X and Ciesielski, V.** 2005. IEEE Congress on Evolutionary Computation. pp. 183–190.
[43] *Classification Strategies for Image Classification in Genetic Programming.* **Smart, W R and Zhang, M.** 2003. Proceeding of Image and Vision Computing NZ International Conference. pp. 402-407.
[44] *Multiclass object classification using genetic pro-gramming.* **Zhang, M and Smart, W.** 2004, Lecture notes in computer science, pp. 367–376.
[45] *Application of Genetic Programming for Multicategory Pattern Classification.* **Kishore, J K, et al.** 2000, IEEE transactions on Eolutionary Computation.
[46] *Representing Classification Problems in Genetic Programming.* **Loveard, T and Ciesielski, V.** 2001. IEEE Congress on Evolutionary Computation. pp. 1070-1077.
[47] *Using Genetic Programming For Multiclass Classification By Simultaneously Solving Component Binary Classification Problems.* **Smart, W and Zhang, M.** 2005, Lecture Notes in Computer Science.
[48] *Issues in Evolving GP based Classifiers for a Pattern Recognition Task.* **Teredesai, A and Govindaraju, V.** 2004. IEEE Congress on Evolutionary Computation. pp. 509-515.
[49] **Bojarczuk, C C, Lopes, H S and Freitas, A A.** Genetic programming for knowledge discovery in chest-pain diagnosis. *IEEE Engineering in Medicine and Biology Magazine.* 2000, pp. 38-44.
[50] *A Novel Approach To Design Classifiers Using GP.* **Muni, D P, Pal, N R and Das, J.** 2004, IEEE Transactions of Evolutionary Computation.
[51] *Discovering Fuzzy Classification Rules with Genetic Programming and Co-Evolution.* **Mendes, R R F, et al.** 2001. Genetic and Evolutionary Computation Conference, Late Breaking Papers.
[52] *Pareto-Co evolutionary Genetic Programming for Problem Decomposition in Multi-Class Classification.* **Lichodzijewski, P and Heywood, M I.** London : s.n., 2007. Proceedings of the 9th annual conference on Genetic and evolutionary computation.
[53] *Integrating Categorical Variables with multiobjective genetic programming for classifier construction.* **Badran, Khalid and Rockett, Peter.** s.l. : Springer-Verlag, 2008. European Conference on Genetic Programming, LNCS.
[54] *Multiobjective classification with mo{GEP}: an application in the network traffic domain.* **Ostaszewski, M, Bouvry, P and Seredynski, F.** Montreal : ACM, 2009. GECCO '09: Proceedings of the 11th Annual conference on Genetic and evolutionary computation.
[55] *Code Growth is Not Caused by Introns.* **Luke, S.** 2000. Genetic and Evolutionary Computation Conference. pp. 228-235. Late Breaking Papers.
[56] *Employing nominal attributes in classification using genetic programming.* **Loveard, T and Ciesielski, V.** singapore : s.n., 2002. 4th Aisa pacific conference on simulated evolution and learning. pp. 487-491.
[57] *Dynamic limits for bloat control in genetic programming and a review of past and current bloat theories.* **Silva, Sara and Costa, E.** 2, 2009, Genetic Programming and Evolvable Machines, Vol. 10.
[58] **Loveard, T.** *Genetic Programming Methods for Classification Problems.* Department of Computer Science, RMI. 2003. PhD Thesis.
[59] *Using enhanced genetic programming techniques for evolving classifiers in the context of medical diagnosis.* **Winkler, S M, Affenzeller, M and Wagner, S.** 2009, Genetic Programming and Evolvable Machines, pp. 111-140.
[60] *DepthLimited Crossover in Genetic Programming for Classifier Evolution.* **Jabeen, H and Baig, A R.** Ulsan, South Korea : s.n., 2009. International Conference on Intelligent Computing.
[61] *Genetic Programming for Medical Classification: A Program Simplification Approach.* **Zhang, M and Wong, P.** 2008, Genetic Programming and Evolvable Machines, pp. 229-255.
[62] *Faster Genetic Programming based on Local Gradient Search of Numeric Leaf Values.* **Topchy, A and Punch, W F.** 2001. Proceedings of the Genetic and Evolutionary Computation Conference. pp. 155-162.
[63] *Genetic Programming with Gradient Descent Search for Multiclass Object Classification.* **Zhang, M and Smart, W.** 2004. 7th European Conference on Genetic Programming, EuroGP. pp. 399-408.
[64] *Learning Weights in Genetic Programs Using Gradient Descent for Object Recognition.* **Zhang, M and Smart, W R.** 2005. Applications of Evolutionary Computing, EvoWorkshops. pp. 417-427.