Advancements in Genetic Programming for Data Classification

by

Hajira Jabeen

A thesis submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy to the FAST National University of Computer & Emerging Sciences

Department of Computer Science

FAST National University of Computer & Emerging Sciences,

Islamabad, Pakistan.

(August 2010)


بسم الله الرحمن الرحيم

Dedicated to my family for their continuous support, patience and love

Acknowledgement

"All praises be to ALLAH" who blessed an ordinary mortal like myself with a part in the vast array of eager knowledge-seekers.

A lot of gratitude I owe to my supervisor Dr. A. Rauf Baig who has been like a spiritual father to me, keeping an eye on my progress and kindly steering my attempts towards success, yielding desired results. His guidance has always been a source of inspiration for me. He wielded his confidence to inculcate my own belief in myself, weighing my ideas and my work in serious regard and putting them at right places. I am also thankful to Dr. Anwar. M. Mirza and Dr. Aftab Maroof for their guidance and full moral support.

I take this opportunity to thank HEC for providing me with financial support and an opportunity to research in FAST-NU. To my colleagues I am obliged for their useful comments on my work, for their fellowship and for creating the dedicated yet cheerful environment that I have been the part of, throughout my research.

I am what I am today because of my family: My parents who did not hesitate to shoulder the responsibilities that were particularly mine going to such lengths not all parents would to make things easier for me, I will never be able to repay you *Ami and Abu.* My grandmother, whose constant prayers are just a bonus on the practical affection and care she has showered upon me, you are simply the best *Dadi Jan*. My mother in law who supported, encouraged and comforted me in every possible way, this is all due to your loving support *Khala.*

My husband who not only encouraged me to pursue this degree but also stood for me and by me in every possible way, if not for him and his all-time support and encouragement, I would never have been able to complete my work. He has been my light in all the dark times, my strength in my frailties. I owe you my life Khuram.

My son, who turned all glooms into sparkling joy with his endless chatter, thanks a lot dear Issmaeel, for accommodating your ever busy mom irrespective of the attention that you deserved.

My special gratitude to my siblings, who provided a light hearted view to serious problems posed by life and kept providing me concurrent review as I progressed along the writing process, I am proud to have you.

Table of Contents

List of Figures

List of Tables

List of Algorithms

Abstract

This thesis aims to advance the state of the art in data classification using Genetic programming (GP). GP is an evolutionary algorithm that has several outstanding features making it ideal for complex problems like data classification. However, it suffers from a few limitations that reduce its significance. This thesis targets at proposing optimal solutions to these GP limitations. The problems covered in this thesis are:

1. Increase in GP tree complexity during evolution that results in long training time.
2. Lack of convergence to a single (optimal) solution.
3. Lack of methodology to handle mixed data-type without type transformation.
4. Search of a better method for multi-class classification.

Through this work, we have proposed a method which achieves significant reduction in bloat for classification task. Moreover, we have presented a Particle Swarm Optimization based hybrid approach to increase performance of GP evolved classifiers. The approach offers better performance in less computational effort. Another approach introduces a new two layered paradigm for mixed type data classification with an added feature that uses data in its original form instead of any transformation or pre-processing. The last but not the least contribution is an efficient binary encoding method for multi-class classification problems. The method involves smaller number of GP evolutions, reducing the computation and suffers from fewer conflicts yielding better results.

All of the proposed methods have been tested and our experiments conclude the efficiency of proposed approaches.

List of Publications

International Journal Accepted Papers

1. Jabeen, H and Baig, A R., "DepthLimited Crossover in Genetic Programming for Classifier Evolution." *Computers in Human Behaviour*, Springer, 2010. (to appear)
2. Jabeen, H and Baig, A R., "Review of Classification using Genetic Programming." *International Journal of Engineering Science and Technology*, Feb 2010, Issue 2, Vol. 2.
3. Jabeen, H and Baig, A R., "A Framework for Optimization of Genetic Programming Evolved Classifier Expressions." *Lecture Notes in Computer Science*, Springer, 2010. (to appear)
4. Jabeen, H and Baig, A R., "CLONAL-GP Framework for Artificial Immune System Inspired Genetic Programming for Classification." *Lecture Notes in Computer Science*, Springer, 2010. (to appear)
5. Jabeen, H and Baig, A. R., "Multi-Class Classification using Genetic Programming." *Lecture Notes in Computer Science*, Springer 2010. (to appear)

International Journal Submitted Papers

1. Jabeen, H and Baig, A. R., "GPSO: Optimization of Genetic Programming Classifier Expressions for Binary Classification using Particle Swarm Optimization." *International Journal of Innovative Computing, Information and Control.* (under review)
2. Jabeen, H and Baig, A. R., "Two Layered Genetic Programming for Mixed Variable Data Classification." *Applied Soft Computing*. (under 2nd review)
3. Jabeen, H and Baig, A. R., "Framework for Optimization of Genetic Programming Evolved Arithmetic Classifier Expressions using Particle Swarm Optimization for Multi-Class Classification."*Knowledge Based Systems*. (under review)

Accepted Book Chapters

1. Jabeen, H and Baig, A R., "Particle Swarm Optimization Based Tuning of Genetic Programming Evolved Classifier Expressions." *Studies in Computational Intelligence (SCI).* Vol 284,pp 385-397. Granada, Spain : Springer, 2010.
2. Jabeen, H and Baig, A. R., "Lazy Learning for Multi-Class Classification using Genetic Programming ." *Communications in Computer and Information Science (CCIS),* Springer 2010. (to appear)

International Conference Accepted Papers

1. Jabeen, H , Jalil, Z and Baig, A R., "Opposition Based Initialization in Particle Swarm Optimization (O-PSO)", Montreal, Canada, 2009, *Genetic and Evolutionary Computation Conference (GECCO 2009).*
2. Jabeen, H and Baig, A R., "DepthLimited Crossover in Genetic Programming for Classifier Evolution", Ulsan, South Korea, 2009, *International Conference on Intelligent Computing (ICIC 2009).*
3. Jabeen, H and Baig, A R., "Particle Swarm Optimization Based Tuning of Genetic Program Evolved Classifier Expressions",Granada, Spain,2010, *International Workshop on Nature Inspired Cooperative Strategies for Optimization(NICSO 2010).*
4. Jabeen, H and Baig, A R., "Framework For Optimization Of Genetic Programming Evolved Classifier Expressions",Sansebastian, Spain,2010. *Hybrid Artificial Intelligent Systems(HAIS 2010).*
5. Jabeen, H and Baig, A R., "CLONAL-GP Framework for Artificial Immune System Inspired Genetic Programming for Classification." Cardiff, UK, 2010. *International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES 2010).*
6. Jabeen, H and Baig, A. R., "Multi-Class Classification using Genetic Programming." Changsha, China, 2010, *International Conference on Intelligent Computing (ICIC 2010).*

7. Jabeen, H and Baig, A. R., “Lazy Learning for Multi-Class Classification using Genetic Programming .” Changsha, China, 2010, *International Conference on Intelligent Computing (ICIC 2010).*
8. Imran, M, Jabeen, H, Ahmad, M, Abbas, Q, Bangyal, W and Abbas, Q “Opposition based PSO and Mutation Operators (OPSO with Power Mutation),” Shanghai, China, 2010, *International Conference on Education Technology and Computer (ICETC 2010).*

Chapter 1. Introduction

1.1 Introduction

Data classification is receiving increasing interest due to its applicability in several real life domains like disease diagnosis, feature recognition, fraud detection and decision making. It is a challenging task because of proliferation, uncertainty and unpredictability of available data. This increases the need of automated classification systems with no or minimum human interference. Over the past years researchers have addressed the problem of using classification systems in many different ways. Some properties of a good classification system are [1]:

- **Applicability:** It should be readily applicable to data in any form, no pre-processing or tuning should be required.
- **Accuracy:** It should be reliable and offer efficient and generic prediction.
- **Efficient modelling:** It should be able to model correct dependencies present in the data, discarding any unnecessary attributes. It should be independent of data distribution and flexible enough to adapt to the data properties.
- **Comprehensibility**: It should be comprehensible to help future decision making.
- **Portability:** It should be portable to other tools for future use and efficacy.
- **Interpretability:** The classification decision should be interpretable.

One of the newly emerged evolutionary algorithms, Genetic Programming (GP) has been found efficient for classification purpose and possesses all of the above mentioned abilities. This important feature of GP has been recognized since its inception and GP has been widely used for the classification tasks since then. While being successful in various application domains, GP suffers from a few limitations like code bloat, long training times, lack of robustness etc. Researchers have been trying to overcome these limitations to make the most out of this powerful classification tool and the same is the focus of this research. This research addresses

the problems associated with the GP and aims at giving a comprehensive solution capable of eliminating these problems without letting go of the benefits in term of classification accuracy. The rest of this chapter explains the background, problem, the motivation and the thesis goals.

1.2 Classification and Evolutionary Algorithms

Our everyday life is infinitely abundant in various classification problems. Man is expert at solving these problems ranging from differentiating between a cup of tea and a glass of juice to more complex tasks like classifying the type of IP packets flowing over the internet. A technically complex classification task can include but is not limited to medical diagnosis, automatic recognition of voice or fault detection in any automated manufacturing process.

The complexities associated with the classification problems and the opportunity to utilize the high processing powers of machines makes the task of automatic classification rather important. Other contributing factors like the lack of human expertise in handling large data, risk of error, associated cost and time also favor the use of automated systems that could perform the classification task. The factors which determine the success of a particular classification methodology are accuracy, robustness, preprocessing requirements and simplicity.

With the amount of processing power available in the form of computers, automated classification has a long and successful history. Dozens of different methodologies have been developed and proven successful for various classification tasks. Statistical classifiers, neural networks, decision trees, rule set classifiers and instance-based-classifiers are some of the major ones. Evolutionary algorithms, being one of the successful methodologies, are the focus of our research.

Evolutionary algorithms (EA) are the stochastic search processes mimicking the behavior of natural evolution proposed by Darwin. A random population of desired solutions is created and evolved using the principle of survival of the fittest. The operators like crossover and mutation are applied during the evolutionary process. Unlike combinatorial algorithms, EAs can search for the optimal solutions without searching the whole solution space. EAs have been found efficient

in performing data classification successfully [1, 2]. There are number of EAs in use today. The two common approaches used in EAs for classification are Michigan approach, where each individual encodes a single prediction rule, and Pittsburgh approach, where an individual encodes a set of prediction rules. The EAs include but are not limited to Genetic algorithms, Differential evolution, Evolutionary strategy and GP. GP holds an edge among other evolutionary algorithms due to its flexibility to model data automatically.

1.3 Genetic Programming for Classification

GP is relatively new and fast developing evolutionary algorithm. It was primarily developed to automatically generate and evolve computer programs. A population of desired random programs is created and evolved in search of a better program. After being successful in the desired domain it has also been applied to handle a variety of other tasks.

GP has gained importance due to flexibility in representation of evolving solutions which can vary in size, shape and length. This important property makes it applicable to the problems where the structure of desired solution is not known. GP has been applied for data classification problem in a variety of unique and interesting ways. This includes the following:

- Evolution of decision trees
- Evolution of classification algorithms
- Evolution of SQL queries
- Evolution of classification logical rules
- Evolution of arithmetic expressions as classifiers

The last method has been mentioned as a GP ‘specific’ method in which an arithmetic classifier expression (ACE) 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.

As mentioned in the previous section, evolutionary algorithms offer a stochastic search that is efficient in finding optimal solution in less time as compared to exhaustive search of solutions. GP is one of the evolutionary search algorithms that enjoys an outstanding position due to several properties already mentioned. GP has been found efficient in solving particular types of problems having following properties [3].

1) Interrelationships of variables is unknown
2) Finding size and shape of solution is part of problem
3) Data is available in computer readable form
4) Conventional mathematical analysis does not provide analytic solutions
5) Approximate solution is acceptable
6) Small improvements in performance can be measured and prized
7) Good simulators to measure the performance of a solution to the problem but poor method to obtain the solution itself

All the above mentioned qualities exist in data classification problem, which makes it feasible to be solved using GP.

1.3.1 Associated Issues

Just like any other algorithm, GP has its downsides too. Despite being an excellent means of solving classification problems, it suffers from the following few limitations.

1) Bloat

During evolution of variable sized programs, average size of the solutions increase. This, in literature, is known as ‘bloat’. The researchers working in GP face this inevitable increase in program complexities during the evolution, increasing the memory and computation time required to evaluate the fitness. This problem increases the size of classifiers as the evolution

proceeds. On the other hand complex classifiers tend to lack generalizing abilities. The details and solutions of this issue will be discussed in Chapter 3.

2) Long Training Time

For the fitness evaluation of an individual during the evolution for data classification, each tree must be evaluated for every instance in the training data increasing the number of function evaluations and in turn the training time required to evolve classifiers. Since time has always been a major resource, researchers have been putting a lot of effort in making this aspect of GP as much time efficient as possible.

3) Lack of Convergence

After the termination of a GP system by either completion of desired number of generations or achievement of certain fitness, the GP system does not necessarily converge to the same solution. The ultimate solutions may differ in structure and fitness. This is known as lack of convergence in GP. This phenomenon is common to all the evolutionary algorithms. A method to increase the efficiency of suboptimal classifiers is required.

4) Multi-Class Classification

GP based classification is binary in nature and it lacks a definitive solution to multi-class classification problems. The two methods currently in use are one class versus all (binary decomposition) method, which requires more number of GP evolutions, and classifier thresholds (real value mapping to different classes), which operates on numerical data only. A generic and computationally efficient method is required to handle this problem.

The aim of this thesis is to address the above mentioned issues and come up with solutions to overcome them.

1.4 Research Goals and Objectives

The major goal of this thesis is to advance the state of the art in the task of data classification using GP. The aim is not to find a classification method that can outperform all the other techniques but to explore the optimal use of GP for data classification.

This main objective can be divided into following four sub-goals.

1.4.1 Reducing Classifier Size

William Ocham , a great logician, presented a principle known as Occam's razor. It states that "Entia non sunt multiplicanda praeter necessitate" or "Entities should not be multiplied unnecessarily". This states that the simplest strategy or explanation is the best one. In science, Occam's razor is used as a heuristic in development of theoretical models. The same principle can be applied to classifiers.

The first goal of the work is to evolve simpler classifiers in anticipation that simpler classifiers would exhibit good generalization abilities. For such a case, a specialized crossover operator that does not let trees increase in their complexity during the evolutionary process and efficiently searches within the predefined limits has been introduced.

1.4.2 Enhancing Classifier Efficiency

The purpose of a classifier is to efficiently predict the class labels of unseen data. Efficient predictions indicate a classifier's better generalizing ability and classification accuracy.

A target of this work is to come up with a method that can enhance the efficiency of classifiers evolved using GP. This optimization should be able to increase classifier accuracy with less computational effort, so that the problem of lack of convergence could be handled with less effort.

1.4.3 Increasing Classifier Robustness

Two different types of classification paradigms exist for different types of data. One is rule based method applicable to nominal data, which can be applied to continuous data after performing discretization. Other is arithmetic expression based method, where the nominal data is converted into numerical values. Both methods must perform a preprocessing step to convert data into the required form; this preprocessing can also lead to biasness in the data. This work aims to develop a robust classification method that can perform classification on the data in its original form.

Some of the above mentioned goals are interdependent, like code bloat adds to the computation time required in GP and compromises the resultant classifier size. Similarly efficiency of a classifier is associated with its size. Simpler classifiers are considered better for their short evaluation time and better generalizing ability.

1.4.4 Multi-Class Classification

Binary or two class classification is a well explored special case of classification problems. Binary classification has been generalized in many ways to handle multiple classes. In case of GP one must spend more computational effort in evolving binary classifiers for different pairs of classes. In case of thresholds, optimal values must be determined for better classification.

We want to come up with a less expensive classification method that can observe better generalizing abilities.

1.5 Research Contributions

In line with the objectives, this research has made the following contributions to the existing literature on the subject.

1.5.1 Taxonomy of Classification Techniques

We have created a taxonomy of approaches that use GP for classification, where each group of methods share same characteristics to tackle the task of classification using GP. This work has been published in

1. Jabeen, H and Baig, A R., "Review of Classification using Genetic Programming." *International Journal of Engineering Science and Technology*, Feb 2010, Issue 2, Vol. 2.

1.5.2 Crossover in GP for Classification

A new crossover operator has been proposed that performs well for classification task without increasing the complexities of programs during GP evolution. This work has been accepted for publication in:

1. Jabeen, H and Baig, A R., "DepthLimited Crossover in Genetic Programming for Classifier Evolution." *Computers in Human Behaviour*, Springer, 2010. (to appear)

1.5.3 PSO based Optimization of Classifiers

A new PSO based optimization technique has been developed that can increase the performance of classifiers with the added advantage of increasing performance with comparatively less number of function evaluations. This work has been accepted in

1. Jabeen, H and Baig, A R., "A Framework for Optimization of Genetic Programming Evolved Classifier Expressions." *Lecture Notes in Computer Science*. (to appear)
2. Jabeen, H and Baig, A R., "Particle Swarm Optimization Based Tuning of Genetic Programming Evolved Classifier Expressions." *Studies in Computational Intelligence.* Granada, Spain : Springer, 2010. Vol 284, pp385-397.

3. Jabeen, H and Baig, A. R., "GPSO: Optimization of Genetic Programming Classifier Expressions for Binary Classification using Particle Swarm Optimization." *International Journal of Innovative Computing, Information and Control.* (under review)
4. Jabeen, H and Baig, A. R., "Optimization of Genetic Programming Evolved Arithmetic Classifier Expressions using Particle Swarm Optimization for Multi-Class Classification." *Knowledge Based Systems.* (under review)

1.5.4 Classification of Mixed Data using GP

A new two layered approach for classification of mixed (real and nominal) data has been introduced. This approach eliminates the need to perform any preprocessing step over the data and deals with the data in its original form yielding good classification results. This work has been submitted in following journal.

1. Jabeen, H and Baig, A. R., "Two Layered Genetic Programming for Mixed Variable Data Classification." *Applied Soft Computing.* (under 2nd review)

1.5.5 Multi-Class Classification using GP

A new binary encoding based multi-class classification technique has been proposed that evolves less number of classifiers as compared to traditional binary decomposition (one class versus all) method. This, in turn, reduces the computational expense and conflicting situations. This method has produced better results when compared to other GP classification methods.

1. Jabeen, H and Baig, A. R., "Multi-Class Classification using Genetic Programming." *Lecture Notes in Computer Science*, Springer 2010. (to appear)
2. Jabeen, H and Baig, A. R., "Lazy Learning for Multi-Class Classification using Genetic Programming ." *Communications in Computer and Information Science (CCIS)*, Springer 2010. (to appear)

1.6 Structure of the Thesis

This thesis is organized into seven chapters as follows.

In Chapter 2, we give a literature overview and present some limitations of current approaches. We attempt to resolve these issues in later chapters.

In Chapter 3, we discuss the problem of bloat and present DepthLimited crossover applied to classifier evolution.

In Chapter 4, we tackle the problem of lack of convergence and long training times and present a PSO based optimization process applied to GP evolved classifier expressions.

In Chapter 5, we present a two layered GP approach to mixed type data classification to handle the problem of lack of convergence.

In Chapter 6, we try to solve the problem of multi-class classification by presenting a binary encoding based classification technique.

In Chapter 7, we conclude the findings of this thesis and propose some avenues for future research.

Chapter 2. Literature Survey

2.1 Introduction

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 complex problems like classification. Its flexible representation has attracted researchers from different fields like bioinformatics, networking and data mining to use GP for their problem handling. GP has been successfully applied to tasks like protein structure or association prediction [4, 5], evolution of network of automata [6], virtual reality and games, time series prediction [7] and symbolic regression [8].

This chapter aims at providing an overview of recent work, relevant to classification, and discusses the advancements made to date. The related issues that need to be addressed for classifier evolution are also mentioned.

2.2 Related Work

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 the frequency of their usage depends upon many things including their simplicity, comprehensibility and accuracy. Techniques like decision trees are simple and comprehensive but applicable to small data sets only; only one tree can be evolved for one set of training samples and they suffer from lack of robustness in presence of noise or missing values [9]. Statistical techniques like bayesian nets, neural networks and support vector machines are complex and results of classification decision are not 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 [10] and Schmidhuber [11]. Later, Koza [12] used the term GP and popularized this technique as a new evolutionary algorithm rather than an extension of GA. GP has emerged as an attractive alternative for classifier evolution. To date, many variations of GP have been introduced to handle various tasks including classification. These variants include Linear GP, Grammar based GP, Graph based GP and Tree based GP [3]. These variants only differ in representations of programs. GP evolves a population of randomly created initial programs and uses a fitness measure to select fitter individuals which take part in further evolution. It works using the Darwinian principle of *survival of the fittest*.

2.3 GP Evolution

The basic GP algorithm is similar to other evolutionary algorithms and works as shown in Algorithm 2-1.

```
Step 1. Begin
Step 2. Initialize population
Step 3. While (desired fitness attained or generations met)
      o   Evaluate fitness of all population
      o   While(new population size)
            ▪ Select parents based on fitness
            ▪ Perform evolutionary operators to create
              children
      o   End while
Step 4. End while
Step 5. End
```

Algorithm 2-1 The GP Evolutionary Algorithm

The added feature of GP is the solution representation and initial construction, which 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 the same primitive set. These randomly

generated solutions are evolved, like other evolutionary algorithms, using the Darwinian theory of evolution to search for better solutions.

The following steps are involved in GP evolution.

2.3.1 Problem Representation

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

1. **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. 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.

2. **Constrained Syntax GP**

   Instead of simple binary trees, the trees might be needed where functions like “if” taking more than two arguments are required. In such trees, some constraints must be placed on the genetic operators to maintain the closure property of the tree after the operator has been applied.

3. **Cellular GP**

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

4. **Linear GP**

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

5. **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.

6. **Grammar based GP**

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

The focus of this thesis is the simple binary tree based representation that is 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 problems.

2.3.2 Solution Initialization

Initialization plays an important role in success of an evolutionary algorithm. A poor initial population can cause any good algorithm to prematurely converge to a suboptimal solution. On the other hand a good initialization leads to genetic diversity in the initial population making most of the algorithms work sufficiently well. There are few initialization techniques popular in tree based GP.

1. **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 and have same depth.

2. **Grow Method**

Grow method randomly selects nodes from function or terminal set and creates random trees till maximum depth minus one is 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.

3. **Ramped Half and Half Method**

Koza [12] 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 [13] and PTC2 initialization [14]. In case of GP, the genetic diversity is important to achieve global optimum value; lack of genetic diversity can lead to premature convergence.

2.3.3 Selection

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

1. **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 also determines the selective pressure; large tournament size favors fitter solutions for selection but causes loss of diversity.

2. **Fitness Proportionate Selection**

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

$$\text{probability}_i = \frac{\text{fitness}_i}{\sum_{j=1}^{n} \text{fitness}_j} \qquad 2\text{-}1$$

   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 [14]. The selection methods contribute to the selective pressure as perceived by Turing that one possibly productive approach to machine intelligence would involve an evolutionary process in which a description of a computer program undergoes progressive modification under the guidance of natural selection (selective pressure or fitness) [3].

2.3.4 Operators

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

1. **Crossover**

   Crossover operator works by selecting two parents from the population. Two random subtrees are selected from each parent and swapped to create children. Some advancement has 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, depth, location or homogeneity of subtrees is also exploited while performing this operation. Details of these types will be discussed in Chapter 3.

2. **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. For example a function node is replaced by a function node of same arity or 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 that node.

3. **Reproduction**

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

2.3.5 Solution Fitness

Fitness is the performance of an individual corresponding to the problem it is aimed to solve. It is compliance of the structure to the task it is required to solve based on a user specified criteria. 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. For example, it can be a measure of error between the original and desired output of a solution or accuracy in case of classification. The difference between fitness evaluation in GP and other evolutionary algorithms is that each individual of GP is a program which must be evaluated in a recursive manner for every training sample. This adds overhead to the algorithm increasing its evolution time and required computational resources.

2.3.6 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.

2.4 Classification using GP

GP has been an area of interest for various researchers since its introduction [12]. It has been applied to solve various problems including data classification for which various techniques have been proposed. In this work we have established a taxonomy of data classification methods based on the approach used to solve the problem. The first approach develops new classification algorithms based on some predefined set of rules that can generate different classification algorithms. The second approach results in logical rules that output a boolean value. These rules are unique in their flexible and comprehensible notations like if-then statements utilizing attributes and their values present in the training data. The third novel approach is evolution of arithmetic expressions that serve as discriminating function between different classes present in training data.

2.4.1 Evolution of Classification Algorithms

GP has been used to evolve classification algorithms like decision trees, fuzzy decision trees, neural networks and rule induction algorithm. For such systems a grammar or a set of rules are predefined. The solutions are initialized using these rules. The structures of solution must 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 operators.

Decision trees are the simple classifiers and GP has been extensively used to evolve them. The work ranges from Koza's explanation [15] to recent [16]. Marmelstein [17] and Bojarczuk [18] used standard GP operators to evolve decision trees using a defined syntax. Folino [19] used a

hybrid GP and simulated annealing system to evolve decision trees. Bot [20] 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 building-block [21] approach, decision trees are built from simpler to complex trees during evolution. Eggermont [22] used 'atomic' representation to represent decision trees. A two layered fitness is used to evolve these trees to prefer smaller trees over larger ones with same accuracy. A parallel GP based approach has been used by Folino using the concept of cellular GP for decision tree evolution [23]. Decision tree evolution methods suffer from the drawbacks of decision trees which are applicable to categorical data only. Their efficiency is disturbed if the training data is too small or too large making decision trees unstable. Moreover a decision tree can become very large requiring further steps for detection and pruning of such inefficient parts.

Tsakonas [24] 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.

Pappa et al [25] defined a set of rules and used grammar based GP to automatically define and evolve rule induction algorithms. 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 [26] has been proposed recently, it can construct solutions, store and update existing solutions by using an adaptive variant of GP. This autonomous system is able to decide if it can solve the problem or not. The proposed system is able to handle classification and regression problems.

Although the evolution of classification algorithms is an innovative idea, it does not overcome the inherent disadvantages suffered by eventually evolved algorithm like neural networks or decision trees. 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. Large grammar tends to introduce more constraints biasing the efficiency of search process.

2.4.2 Evolution of Classification Rules

Classifier rules are usually expressed 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 [27]. However, individual rules can also be learned from training data. GP has been used for evolution of classification rules since long [28]. 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 the consequence of the rule is the resultant class.

Frietas [29] 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 [30] introduced interesting and comprehensive ‘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 [31] used GP to evolve rules 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 [32] used GP to evolve comprehensible simple rules for continuous attributes by combining the parallel searching ability of GP. 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 [33] 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 [34], Berlanga [35] and Mendes [36] introduced evolution of fuzzy rules using GP. Chien [37] used fuzzy discrimination function for classification. Bozarczuk [38, 39] 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 [40] introduced two GP based systems for medical domains and achieved noticeable performance. Lin [41] proposed a layered GP where different layers correspond to different populations performing the task of feature extraction and classification. Various other rule-based classification methods have also been introduced [42-44].

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

2.4.3 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 to assign thresholds. The method includes assigning static thresholds [45, 46], dynamic thresholds [46, 47] and slotted thresholds [48]. Another scheme is binary decomposition. In this technique a classifier for each class is evolved separately considering all the other classes of data as a single 'not desired' class. All the resulting best classifiers are integrated into one final classifier. Classification decision is made based on outputs of aggregated classifiers. The classifier with positive output or maximum output is declared winner. Binary decomposition methods have been explored in many research works [49-53]. A relatively different, GA inspired, method for multi-class classification has been proposed by Durga [54], an amalgamated chromosome (vector) of classifiers for all the classes is evolved in single GP run. Other effective multi-class classification methods include Mende's work [55] 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 [50] 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 multi-class classification.

2.5 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 data imbalance [49]. 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 [29]. The researchers have also used more than one fitness measures for classifier evolution task [56]. 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.

2.6 Multiobjective Genetic Programming

This technique cannot be classified as one of the special technique for classification. This technique can be and has been used to evolve classifiers using any of the above mentioned types. The main idea behind multiobjective optimization is to have more than one fitness function for each population member and a desire to optimize the solutions for each fitness function. The solution must be acceptable according to all the fitness functions simultaneously. This technique is popular in GP to favor simpler solutions because the increase in program sizes during evolution is a major drawback of GP. In case of classification a simple and accurate classifier is desired. Therefore common objectives that are taken into account in most of the cases are classifier size and accuracy. Lichodzijewski [57] 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 Badran's work [58] two objectives were taken into account; number of nodes in a tree and misclassification error over the training set, for the classification of nominal data. In another work for network traffic, classification has been performed by Ostaszewski [56] where the objective functions are sensitivity and specificity. The classifiers yielded high performance making it applicable for network security problems.

2.7 Strengths and Weaknesses of GP Classification

2.7.1 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. Such mechanisms are not prune to local optimum values. 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 common problem solutions. This eliminates the need of having different genotype to phenotype encodings. 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 does not need any explicit information or preprocessing regarding class or attribute dependencies.

GP can automatically eliminate attributes unnecessary for the classification performing the task of feature extraction algorithm [54]. Similarly important attributes can appear near the root whereas less important ones would appear deeper in the tree [59].

The classifiers obtained through GP are usually comprehensive and transparent. 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 [49].

GP evolves the classifier in the form of a program. The final classifier program should determine 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 [21].

The classifier representations differ in each separate execution, so several different classifiers with same or slightly different accuracy can be extracted [49]. This is also called lack of population convergence. Although undesirable in a few cases, it can search variety of solutions for the same problem.

GP has the ability to operate upon the data in its original form [49]. No preprocessing or data transformations are necessary to apply GP for classification task. Yet, type conversions might be needed, for a special function set to be applicable on all the data or to make the data set contain same type of attributes, depending upon the primitive set and the method being adopted for classification [58].

GP based classifier evolution is not affected by the data distribution. 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 [60].

Some of the above mentioned benefits have also been reported by Poli [3], 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 is measureable and availability of simulators to measure the performance of solutions but poor methods to obtain the solution itself. This can be observed that all these properties are inherent in the problem of classification, making it a feasible application domain for GP. All these factors count for preference of GP for classification problems. Numerous

researchers have applied GP for the said task and loads of work done in the field can be found. The missing item in all this work is a meaningful categorization and analysis of these techniques.

2.7.2 Weaknesses

In the previous section numerous advantages of using GP for classification has been discussed. 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 little attention in the classification scenario in the past. Few researchers have attempted to tackle these individual problems recently but a definite solution has yet to be found.

The drawback of GP is the necessity of frequent evaluation of fitness of each program in the population in each generation. In case of classification the number of training samples is usually large, and a classifier must be recursively evaluated for all training instances. In case of 'N' population size, 'E' generations and 'T' training instances, the number of fitness evaluations would be 'N * E * T'. This makes the fitness evaluation of an individual, the most time consuming operation of the algorithm [61].

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 heavy 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 [18].

GP based classifiers are either applicable to nominal or numerical data. For both types of data to work, conversion from one type to another must be performed which acts as a possible source of bias in addition to extra computation. 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 to 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 classification of data having more than one class labels, this is also known as multi-class classification. Several methods for multi-class classification have been proposed but the technique lacks a definite solution.

2.8 Possible Solutions

The focus of this section is to propose 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 yet 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 have been tackled by using some efficient search strategy, one example of such search mechanisms is *"pyramid search"* proposed by Loveard [62]. 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. Reduction of population size risks the lack of genetic diversity during evolution. Some other intelligent search methods can be devised to avoid training time. Besides reducing the population size, training samples can also be reduced to decrease the training time. A similar method named 'incremental learning' has been used by Muni [54] and Kishore [49].

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 [63] used tree size limits with intron detection and pruning, Muni [54] and Kishore [49] used tree size limits. Eggermont [22] used two layered fitness. Depthdependent [64] crossover was used by Badran [58]. But none of these methods have been designed for classifier evolution, or compared with traditional GP in case of classification. There is a need for introduction of specialized crossover that does not let classifiers increase in size unnecessarily as larger classifier compromises the generalizing ability of a classifier. Zhang [66] used an approach that simplifies algebraic expressions by applying reduction formulas using a hashing method. Other methods include tree complexity penalty in fitness evaluation (parsimony pressure) [3], limits on

crossover operations like size fair crossover operator used for GP based classifier evolution [63] have been used for classifier evolution and limits on maximum tree size [54].

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 [67] performs gradient search for optimization of ephemeral constants or numeric constant values present in GP trees producing better results for symbolic regression. Zhang [68] 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 online learning is similar to incremental learning in Neural Networks and offline learning is similar to batch training in neural networks. The gradient descent algorithm is augmented to the existing GP system and applied on each program of the population in a given generation. The remaining evolutionary process remains conventional. The results conclude that online scheme offers better performance. In another motivating work [69], 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 with GP, increasing the complexity of an already computationally extensive task, although considerable increase in performance has also been achieved.

The applicability problem has been handled by Loveard [70] 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 [58] 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 multiple classes, one of the proposed methods is to assign thresholds. The thresholds could be static [45, 46], dynamic [46, 47] and slotted [48]. The problem with this method is that it is applicable to the expressions that output real value rather than boolean value. Another scheme for multi-class classification is binary decomposition; a classifier for each class is evolved separately considering other the other classes of data as single ‘not desired’ class. All the resulting best classifiers are integrated into one final classifier. The classification decision is

based on outputs of all classifiers. The classifier with positive output or maximum output is declared winner. Binary decomposition methods have been explored in [49-54] various research contributions. The major drawback of both the approaches is the conflict between more than one classifier that needs careful handling.

In this research, all these issues have been addressed and an optimized solution for each one of them has been proposed.

2.9 Conclusion

GP is an interesting technique for data classification and it has been noted that GP can perform the task of classifier evolution effectively. It has achieved compatible or better performance in many instances. However, GP based classifier evolution suffers from several problems like long training time, bloat and lack of convergence. There is a need to optimize the process of classifier evolution using GP. The present classification techniques lack robustness, means of decreasing the training time, making the classifiers bloat free and any mechanism to overcome the problem of lack of convergence.

Chapter 3. DepthLimited Crossover

3.1 Introduction

In this chapter code bloat, which is one of the major problems faced by GP, has been focused upon. Code bloat is the inevitable increase in an average individual's size and complexity during evolution due to flexibility to evolve variable length solutions. In case of classifier evolution, this increase results in larger classifiers, reduces their generalizing abilities and increases the training time. This problem has attracted numerous researchers to evaluate the cause of such behavior and search for preventive measures. Previous work lacks focus on classification and classifier evolution tasks. In this chapter, a crossover technique has been introduced that reduces bloat and favors evolution of simpler classifiers. The performance of proposed crossover technique has been evaluated for various benchmark classification datasets. In our experiments, the proposed DepthLimited crossover technique outperforms GP without depth limitation in terms of accuracy, simplicity of classifiers and smaller average population complexity during the evolution. The results obtained using DepthLimited GP (DLGP), are comparable to several other GP based classification methods.

In this chapter, various theories explaining the causes of bloat have been reviewed. Several solutions to tackle and avoid bloat are present in the literature. The crossover techniques, that either try to avoid bloat or favor fitter solutions during evolution have been focused in this work. DepthLimited crossover operator is presented after literature review. The validity of proposed operator has been tested for various benchmark classification problems presented in the results section. The last section concludes the findings.

3.2 Related Work

Bloat is one of the well known and widely studied problems in GP. It offers several disadvantages making it undesirable and increasing the need for its elimination. These drawbacks include increase in training time, stagnation of population, inefficient computations and bloat driven larger solutions. Many researchers have explored the causes of occurrence of bloat and many have proposed solutions to avoid bloat. This section details some of the common bloat theories and corresponding proposed solutions that use crossover.

3.2.1 Bloat Theories

Fitness Causes Bloat

This theory states that there are many different ways to represent the same program especially in the case of variable-length representations. There are more possible ways to represent a solution by its larger counterparts than by smaller ones. The evaluation function assigns the same fitness to all the representations as long as their behavior is the same. On the other hand, as a consequence of destructive crossover, good solutions start decreasing and the selection process gets biased towards the programs having same fitness as their parents. Since larger programs with same fitness are more in number, the evolution process favors larger trees causing bloat. The selection in evolutionary process favors trees with better fitness, this searches for alternative representations of same solutions causing bloat. This is why the theory is named "fitness causes bloat". This theory was first presented by Langdon and Poli [71]. This has also been named solution distribution [72], diffusion [59], drift [73], nature of search spaces [74] and entropy random walk [75].

Protection Theory

This theory [73] is also known as replication accuracy [76] and intron theory [77]. It is based upon the fact that the crossover in GP tends to be more destructive than constructive. This implies that the crossover that exchanges inoperative code will have

less destructive effects. The trees having inoperative code will survive the crossover while maintaining their fitness and will be favored during evolution, increasing average tree size without corresponding increase in fitness.

Removal Bias Theory

In the removal bias theory, Soule [78] blames the crossover for code bloat in GP. He suggests that presence of introns provide regions where addition or removal of subtrees does not affect its fitness. For removal of a subtree they must be contained within an intron, however they may or may not lie deep in a tree. On the other hand, addition of a branch inside inviable code does not affect fitness, regardless of its depth. This asymmetry of addition and deletion of code causes code growth.

Depth Correlation Theory

Luke [59] observed that the depth of a subtree is correlated to fitness. For a node change in tree, the deeper the node the smaller its effect on constituent tree's fitness. Therefore, larger trees will benefit from selective advantage. This theory is considered as a generalization of removal bias theory.

Crossover Bias theory

Dignum [79] explains that the reason for increase in average number of nodes is that crossover in itself adds or removes same amount of genetic material so it does not produce growth or shrinkage. Crossover pushes population towards Lagrange distribution of second kind, where smaller programs have higher frequency. The programs of small sizes are usually less fit which gives selective advantage to trees above average size, increasing mean program size.

HitchHiking Theory

Tacket [80] introduced another phenomenon of code increase. He explains that introns act as hitchhikers and become attached to subtrees that exhibit good traits, therefore a recombination operator is more likely to exchange good subtrees along with hitchhikers. He also showed that the code growth is proportional to the selective pressure and the only time when bloat does not occur is when fitness function is totally rejected during the evolution.

3.2.2 Crossover: The Culprit

It can be observed that almost all the above mentioned theories blame crossover for the existence and prevalence of bloat. This can be due to its destructive nature, or introns. This implies that an efficient crossover mechanism must be developed that does not favor increase in tree size. Several crossover methods designed to avoid excessive increase in tree size during GP evolution, have been proposed in literature. Few of the crossover techniques have been developed to increase efficiency of search and avoid the destructive effects faced by crossover. These crossover techniques will be reviewed in detail in the following section.

3.2.3 Crossover Operators

Numerous bloat control methods have been proposed in the literature. In this section, the methods that attempt to restrict the code growth during crossover operation and the techniques that increase the effectiveness of GP evolution have been discussed.

Size Fair Crossover

Langdon [13] presented an efficient method to avoid code bloat “size fair crossover”. A subtree is selected randomly from one parent; second 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 proven

effectiveness in terms 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, the method is same except that 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 Preserving Crossover

In 'strong context preserving crossover' [81], nodes are labeled based on their coordinates and nodes with similar coordinates were allowed to be swapped. In case of 'week context preserving crossover', any subtree rooted at same nodes with same coordinates could be swapped.

Context Aware Crossover

Context aware crossover [82] attempts to preserve context of the subtrees being swapped. It selects the nodes by comparing and selecting most similar subtrees, to swap, in crossover operation.

Brood Recombination Crossover

A fitness conscious crossover is presented by Tackett who named it 'Brood Recombination Crossover' [80]. A brood 'n' is created and the process of crossover between two parents is repeated 'n' times ,i.e. 2n children are created and 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 hence the only emphasis is on the fitness of resulting trees.

Depth Dependent Crossover

A technique named ‘Depth Dependent Crossover’ was proposed in [64]. 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 [64].This method uses a user defined parameter ‘depth probabilities’, which in later work, is adjusted using a selftuning mechanism. Both methods make no effort to reduce tree complexities during evolution, as they try to secure larger building blocks during evolution.

3.2.4 Ideal Crossover Characteristics

Hammad[83] has defined some salient properties of an ideal crossover. An ideal crossover should be:

- Constructive and it should frequently create children better than their parents.
- Able to force population to converge to a better space of solution yet able to maintain diversity for exploitation.
- It should not favour increase in tree sizes or occurrence of bloat.
- Applicable to all kinds of problem solution and should not involve extra computation.
- It should be easy to implement.

3.3 Proposed DepthLimited Crossover

This section explains the basic steps that make up the methodology of proposed GP based classification algorithm.

3.3.1 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 [12] 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 [49, 54].

3.3.2 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 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 value will be perfect for all the datasets. The classifier represents relationships between attributes present in data, it would be intuitive to define a maximum depth that should include all the attributes present in the data. Let A_d be the number of attributes present in the data then such initial depth can be

$$depth_p = ceil(\log_2(A_d)) \quad 3\text{-}1$$

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.

$$depth_p = ceil(\log_2(23)) = 5 \quad 3\text{-}2$$

A full tree with depth five has 32 leaf nodes. And such a tree can possibly contain all the twenty three attributes present in the data under consideration. As already mentioned in the previous chapters, 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 which implies that it has more importance. Therefore $depth_p$ has been used as maximum allowed depth for ramped half and half initialization in our approach.

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)
- Choose a random subtree S_2 from P_2
- Calculate depth of subtree DS_2
- Choose a random subtree S_1 from P_1 such that DS_1 =DS_2
- Swap S_1 and S_2 to create two children C_1 and C_2

Step 5. Else
- Choose a random subtree S_1 from P_1
- Calculate depth of subtree DS_1
- Choose a random subtree S_2 from P_2 such that DS_2 =DS_1
- 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

Algorithm 3-1 DepthLimited Crossover for Classfication

3.3.3 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 [49]. 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 having smaller depth. This is to ensure that the second subtree with same depth exists in the other parent. The only restriction

placed on the second subtree is that it should have same depth as the first subtree. The algorithm is explained in Algorithm 3-1 and illustrated in Figure 3.1.

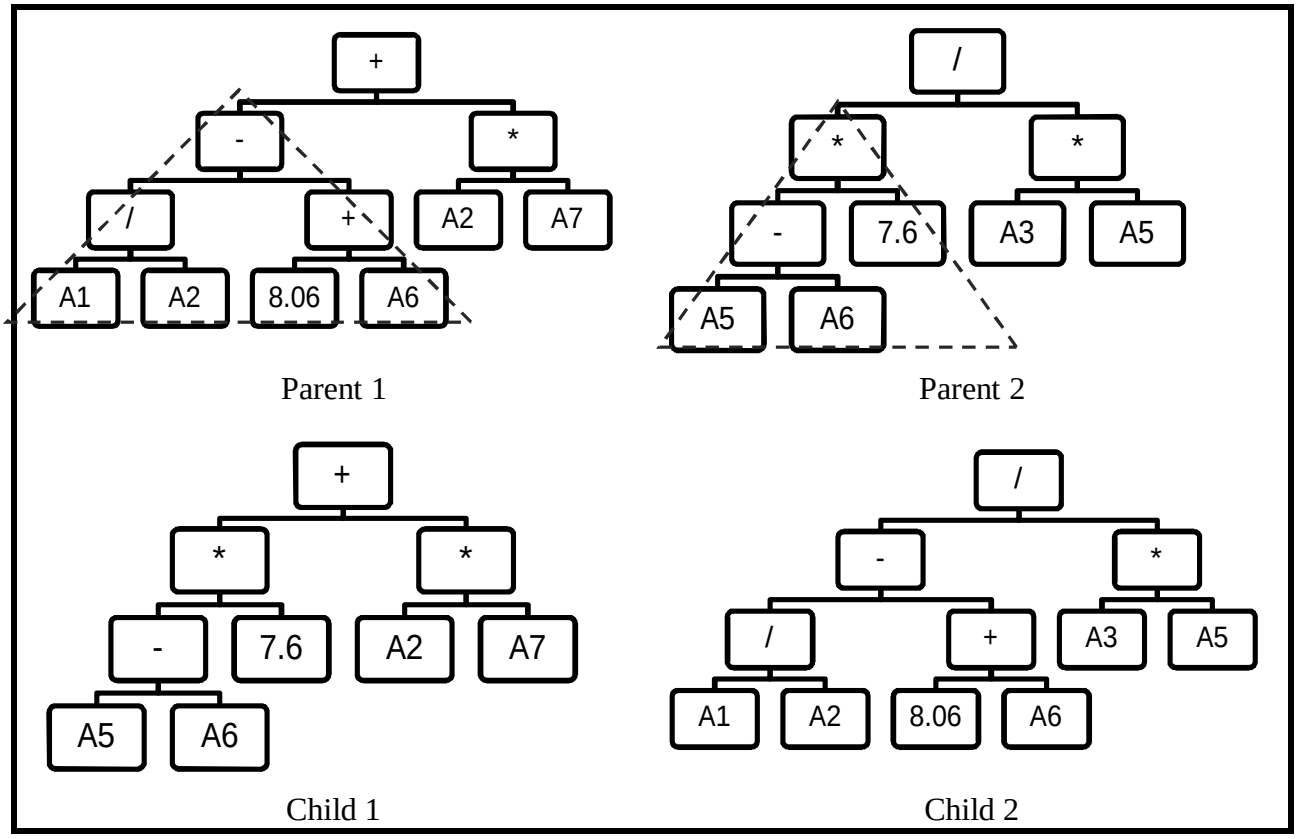

Figure 3-1 DepthLimited crossover operator

3.3.4 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 [30, 54]. 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 3-2.

```
Step 1. Begin
Step 2. int count_correct=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.  Count_correct++
        c. if Value<0 and class = not desired class
             i.  count_correct++
        d. End if
Step 4. End for
Step 5. Fitness=(count_correct/N)*100
Step 6. End
```

Algorithm 3-2 Fitness for Classification

3.3.5 Classifier Evolution Algorithm

The detailed description of the classifier evolution algorithm is given in Algorithm 3-3. This method has been proved efficient for data classification and has been used by several researchers with little modifications [49, 50, 53]. The output of this phase is an ACE 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.

```
Step 1. Begin
Step 2. gen = 0, number of generations elapsed
Step 3. fit_g = 0, best fitness found till the generation gen
Step 4. Initialize generations to user defined value
Step 5. Select one class as 'desired' and other 'not desired' class
Step 6. Initialize GP-ACE population using ramped half and half method
Step 7. While (gen <= generations or Fit_g= 100 )
        a. Evaluate fitness of each member in population (Algorithm
           Fitness)
        b. Find best in population and update Ace_Best
        c. Fit_g=fitness(Ace_Best)
        d. Perform evolutionary operators (Algorithm GP Evolution )
        e. gen = gen + 1;
Step 8. End while
Step 9. Output ACE_Best as classifier
Step 10. End
```

Algorithm 3-3 GP Algorithm for Classification

3.4 Results

Twelve benchmark binary classification problems have been used in this work. Eleven problems are taken from UCI ML repository [84]. One of the dataset named Ripley's data has been taken from [85]. We have selected the datasets based on following properties

1) Dataset should be real or numerical valued
2) Problem should be binary classification
3) Number of instances should not be too much or too less
4) Number of attributes should not be too much
5) There should be no missing values

The datasets have been chosen from various dimensions of life to show the applicability of GP classification as well as generalization of our proposed optimization 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 [54]. Similarly Statlog (Heart) 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.

Table 3.1 Datasets used for experimentation

Datasets	**Referred as**	**Attributes**	**Type**	**Instances**
Breast Cancer Wisconsin	WBC	9	Integer	699
Liver Disorders	BUPA	6	Categorical, Integer, Real	345
Haberman's Survival	HABER	3	Integer	306
Parkinsons	PARK	22	Real	197
Pima Indians Diabetes	PIMA	8	Integer, Real	768
Blood Transfusion Service Center	TRANS	4	Real	748
Ionosphere	ION	34	Integer, Real	351
SPECTF Heart	SPEC	44	Integer	267
Ripley's data	RIPR	3	Real	1250
Connectionist Bench	SONAR	60	Real	208
Musk (Version 1)	MUSK	166	Integer	476
Statlog (Heart)	HRT	13	Categorical , Real	270

3.4.1 Experimental Setup

The parameters used for classifier evolution using GP are mentioned in Table 3.2. These parameters have been selected after empirical analysis of the problem and found efficient for the solution of the underlying technique.

Table 3.2 GP parameters used for classification

Population	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 and 6 for GP with no depth limits
Function Set	+,-,*,/ (protected division,division by zero is zero)
Terminals	Data Attributes A_1,A_2…A_n, Ephemeral Constant [0,10]
Termination Criteria	User specified generations or 100% training accuracy of classifier

Initial maximum depth limit

To test the validity of our proposed initial max depth, experiments are performed using five datasets having different number of attributes and instances. As presented in Table 3.3, the $depth_p$ initial depth has performed better making it a feasible initial maximum depth value. Therefore we have used $depth_p$ initial depth limits for other datasets.

Table 3.3 Analysis of various initial depth limits

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%

After definition of search limits, the results of proposed approach are presented. Three measures have been used to show effectiveness of the proposed approach

1) Classification Accuracy
2) Code Complexity
3) Classifier Simplicity

All the reported results are averaged for ten executions of tenfold cross validation for each dataset with five different partitioning of the data.

3.4.2 Classification Accuracy

Comparison with GP (without depth limitation)

Table 3.4 summarizes the 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 with application of DepthLimited crossover.

Table 3.4 Comparison with GP without Depth Limitation

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

It is evident that the difference in performance of both algorithms is not very prominent, but the gain in terms of simplicity of classifiers and that of bloat is manifold.

Comparison with other GP approaches

Now we will 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 3.5 Comparison with other GP based techniques

Datasets	**DepthLimited crossover GP**	**Others**
WBC	96.6%	95.1%, MultiTree GP [54], 96.4%, Dynamic Range GP [50], 93.92-97.54%, Rule GP [32] , 97.9%, Enhanced GP [63], 98.2%, Simplification GP [66], 92.5-96.24% GP structures [24]
BUPA	69.2%	68.4%, Dynamic Range GP [50], 60.8%, MultiTree GP [54]
PIMA	68.6%	74.2%, Dynamic Range GP [50], 68.3-75.75% GP structures [24], 68.64-75.16% Rule GP [32] , 74.87-75.53% Enhanced GP [63]
ION	88.5%	85.4-90.52% Rule induction [25]
SPEC	77.6%	83.2±7.3, Simplification GP [66]
SONAR	73.3%	68.4-72.42% Rule induction [25]
HRT	88.3%	71.4-79.66% GP structures [24], 78.01-92.81% Rule GP [32] , 74.2-77.9% Rule induction [25]

It is apparent from Table 3.5 that the achieved results are compatible with other GP based techniques present in the literature; making our technique effective in reducing complexities. This is due to the fact that proposed crossover technique efficiently searches through the space

and does not spend unnecessary computations searching along infeasible lengths of solutions, making it simpler and efficient.

3.4.3 Code Complexities

$Depth_p$ is the maximum depth defined for each dataset the depths for each dataset are mentioned in Table 3.6. as we can see that the trees with $depth_p$ depth are capable to contain all the attributes of the data.

Table 3.6 Maximum allowed depth for each dataset

Dataset	**Attributes**	**$depth_p$**	**Maximum Nodes**
WBC	9	4	31
BUPA	6	3	15
HABER	3	2	7
PARK	22	5	63
PIMA	8	3	15
TRANS	4	2	7
ION	34	6	125
SPEC	44	6	125
RIPR	3	2	7
SONAR	60	6	125
MUSK	166	8	511
HRT	13	4	63

Figure 3-2 and Figure 3-3 show the average population size during GP evolution using standard GP and DepthLimited crossover. Comparison of number of average nodes in case of DLGP and GP with no depth limits discloses a noticeable difference in average number of nodes. In case of traditional GP where no restrictions have been placed on evolving population size its complexity increases unlimitedly. The number of nodes per tree has reached up to 5000 in the worst case. On the other hand, for DLGP 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. The maximum number of

nodes attained by any dataset using DepthLimited GP has not exceeded the maximum nodes mentioned in Table 3.6.

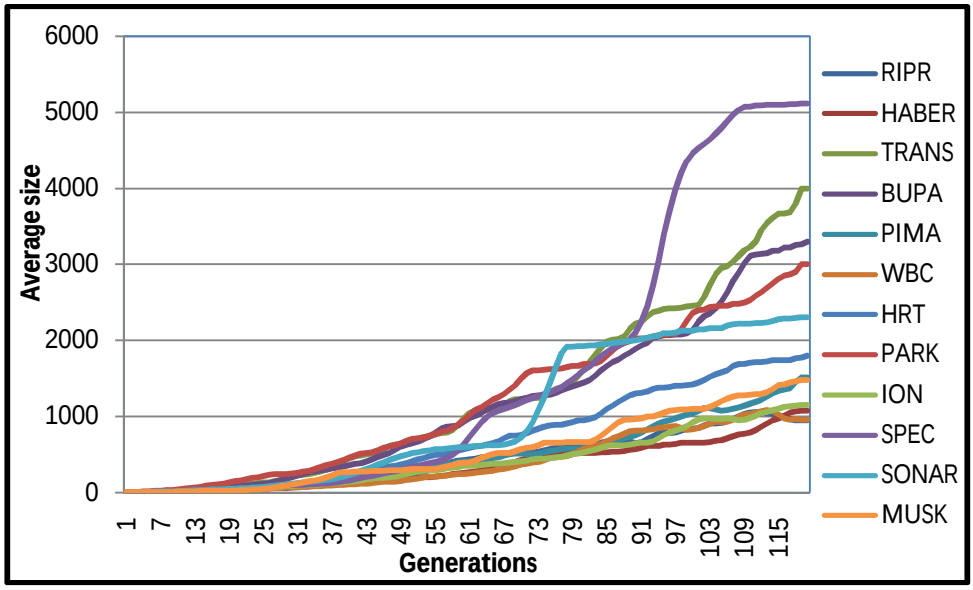

Figure 3-2 Increase in average population size using GP with no size limits

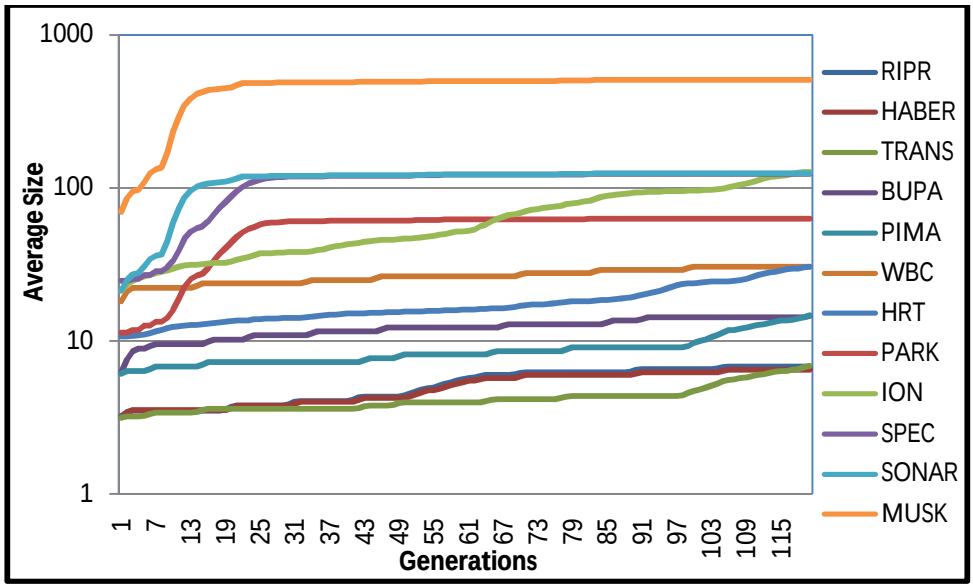

Figure 3-3 Increase in average population size using DepthLimited GP

3.4.4 Simplicity of Classifiers

Figure 3-4 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. 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 DLGP method; indicating immense improvement in terms of simplicity.

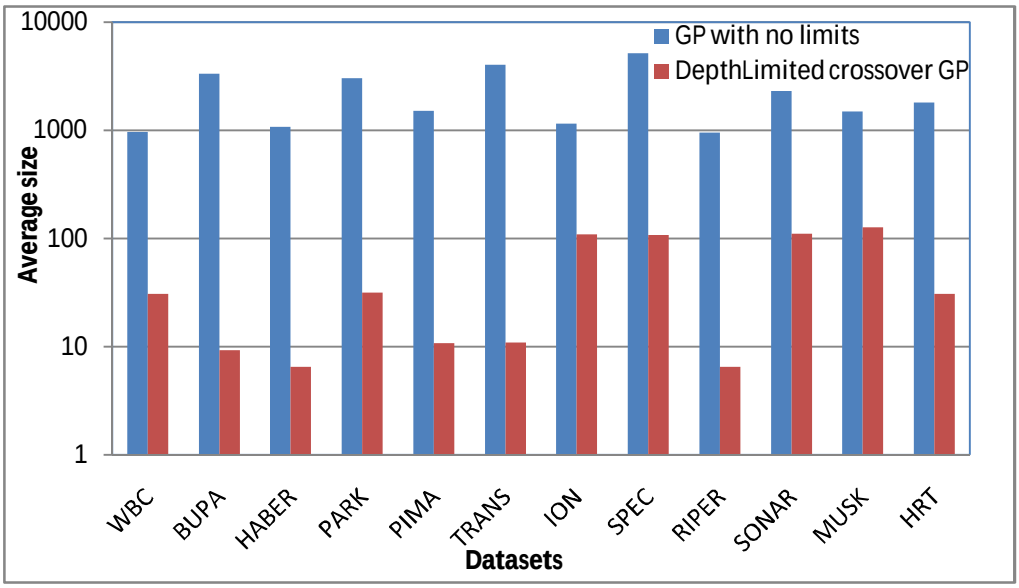

Figure 3-4 Average number of nodes in classifiers

3.5 Conclusion

Based on our rigorous testing detailed earlier, the DepthLimited Crossover has proved to be highly efficient in terms of computational resource utilization and simplicity of classification. It has yielded totally compatible results making it a superior version of its predecessor. This procedure, however, warrants very careful approach while defining the initial limits so that GP can efficiently search in this predefined space. Once liberal initial limits are defined, it can efficiently search for desired solution in that space yielding compatible results.

Chapter 4. Optimization of GP Evolved Classifiers

4.1 Introduction

GP classification techniques suffer from certain limitations already highlighted in the earlier chapters. Two of these issues will be addressed in this chapter. The first problem is the long training time and increase in complexity of population during the GP evolution. The second problem is that GP yields different results after each execution both in structure and performance. This is also known as lack of convergence in GP. Both of these problems are well reported in the literature [3]. In this chapter, a strategy which aims to overcome the above mentioned limitations is proposed. It is demonstrated that the efficiency of classifiers can be increased by optimizing the evolved classifier expressions. Particle Swarm Optimization (PSO) algorithm has been used for the optimization purpose.

PSO is a population based, direct, stochastic optimization method proposed by Kennedy and Eberhart [86]. The algorithm simulates the behavior of bird flock flying in a multidimensional space in search of some food. While doing so, birds adjust their movements according to the constraints of the environment. The flight of these birds is attracted towards their previous best positions and the bird which is nearest to the food. These values can be called local and global best positions. All the birds continue their movements till all of them reach the best position in other words converge to a single solution. PSO is popular due to its simplicity, comprehensibility, less vulnerability to local optimal values and faster convergence. Another reason for efficiency of PSO is that there are very few parameters to adjust. PSO has been proved successful in a wide range of real life problems, particularly for parameter optimization in continuous, manifold search spaces.

The contributions of this work are a GP and PSO hybrid algorithm (GPSO) for classifier evolution. In GPSO the accuracy of GP evolved ACE is increased by PSO after addition of weights. This better accuracy is achieved in less NFE as compared to GP alone approach.

The performance of proposed GPSO Hybrid algorithm is tested on several binary classification datasets from UCI and found efficient in all the classification problems.

4.2 Related Work

In this section we discuss data classification methods that use GP and some optimization techniques for GP expressions followed by an overview of PSO algorithm.

Optimization of GP expressions can be achieved by simplification and reduction in size of GP trees. Such methods have been proposed by several researchers. Zhang [66] has emphasized evolution of simple classifier expressions using GP. He used an online simplification approach. The algebraic expressions are reduced using a hashing method. This method has been found efficient in simplification of evolved classifier and reducing the time required to evolve complex structures. Other methods include tree complexity penalty in fitness evaluation (parsimony pressure) [3], limits on crossover operations like size fair crossover [13], operator used for classifier evolution [63], or DepthLimited crossover [65] and limits on maximum tree size [54].

Relatively less work has been done to optimize GP evolved intelligent structures. A distinct method proposed by Topchy [67] performs gradient search for optimization of ephemeral constants or numeric constant values present in GP trees producing better results for symbolic regression. Zhang [68] 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. In another motivating work [69], weights are added to all the edges present in a GP expression tree and optimized using gradient descent. The method was found efficient in terms of accuracy. Both methods proposed by Zhang are successful but they are added to the evolutionary process adding the computational effort into an already resource intensive task. This raises the need for an optimization algorithm that is computationally efficient and yields better performance.

Different optimization algorithms (Gradient Descent, Back Propagation, Particle Swarm Optimization, and Genetic Algorithms) have been proposed in recent years and are being efficiently used for function optimization. PSO has been found efficient for optimization tasks and offers several advantages. It works using a simple concept and is easy to implement. It is computationally efficient and has been found effective in a wide variety of applications like function optimization [87], neural network training [88] and fuzzy system control. PSO solves optimization problems by simulating bird flock/swarm flying together in n-dimensional space in search of some optimum place, adjusting their movements in the constrained environment. Considerable research has been done for optimization and efficient working of PSO. Several parameters have been introduced to improve the performance of PSO. Two important parameters are constriction coefficients and inertia weight. Constriction coefficients set the proportion, with which the previous best position of a particle and the global best particle of the swarm attract the movement of a particular particle. A PSO variant with a constriction factor has been introduced by Clerc [89] who analyzed the convergence behavior of the PSO. Constriction factor guarantees the convergence and improves the exploration ability of swarm. The inertia weight determines the step size for movement. Shi [90] introduced the concept of linearly decreasing inertia weight which is much like the temperature in the simulated annealing context or like the fluidity of the medium. A fuzzy scheme to nonlinearly change the inertia weight is proposed by Shi [91].

To improve the performance of PSO, another research focus has been the variations in PSO topology. Keneddy [92] proposed that PSO with smaller neighborhood performs better in complex problems and larger neighborhood would perform better in simpler problems. Suganthan [93] suggests dynamically increasing neighborhood, until it includes all the particles of the swarm. Parsopoulos uses a combination of the global version and local version to make a unified particle swarm optimizer (UPSO) [94]. Mendes introduce a fully informed PSO [95], in which all the neighbors of the particle are used to update the velocity. The influence of each particle on its neighbors is weighted based on its fitness value and the neighborhood size. Some variations in basic behavior of PSO have been proposed, i.e. attractive repulsive PSO (ARPSO) [96], predator prey approach to PSO (PPO) [97], species based PSO (SPSO) [98] and charged swarm [99]. The purpose of all these variations is to avoid premature convergence, working for multimodal functions to find multiple optima.

The factor that makes this method unique is the use of PSO for optimization of GP evolved expressions. Such an instance has not been found in the literature. Next section explains the proposed optimization process followed by the classification algorithm.

4.3 Proposed Hybrid GPSO

This section presents the approach used to perform the data classification and optimization of evolved classifiers. Figure 4-1 gives an overview of the proposed GPSO algorithm.

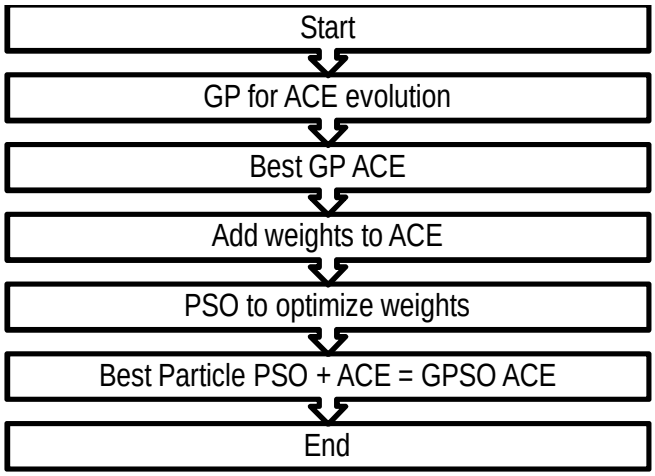

Figure 4-1 GPSO hybrid algorithm for classifier optimization

The first step of proposed algorithm is to evolve ACE using GP. These evolved ACE are then prepared for tuning by addition of weights along each terminal. These weights are optimized using PSO. The detailed description of GPSO is given in the Algorithm 4-1 illustrated below.

The proposed technique involves three phases:

1) GP phase for the evolution of ACE
2) Preparation for optimization phase by adding weights to the best found ACE
3) PSO phase for the optimization of newly added weights

```
Step 1.  BEGIN
Step 2.  Apply GP to evolve ACE (Algorithm GP Classification)
Step 3.  Add weights to the best ACE.
Step 4.  Initialize all weight particles.
Step 5.  Optimize these weights using PSO.
Step 6.  Output Best GPSO-ACE
Step 7.  END
```

Algorithm 4-1 Hybrid GPSO for Classification

The description of each phase is as follows:

4.3.1 GP Phase

The first phase of the proposed algorithm is to evolve ACE using GP. For this purpose we have used DepthLimited crossover based GP explained in chapter 2. However, some other variant can also be used to evolve the ACE. The advantages of using DepthLimited crossover is that the classifier trees are of moderate size and the classifier size cannot exceed beyond maximum defined depth. Let A_d be the number of attributes present in the data then maximum initial depth limit defined for depth limited GP is

$$depth_p = ceil(\log_2(A_d)) \quad 4\text{-}1$$

For a tree of $depth_p$ the maximum number of possible terminal nodes, $Nodes_{max}$ are

$$Nodes_{max} = 2^{depthp} \quad 4\text{-}2$$

The best ACE from the GP evolution phase with the bounds described above is returned to the PSO based optimization phase. This bound on maximum number of possible nodes also implies a bound on maximum number of dimensions of a particle for PSO optimization. This also saves PSO from suffering from the “curse of dimensionality”. The introduction of this term goes back to 1957 when Bellman introduced this term. This problem has been reported to affect the PSO performance.

4.3.2 Preparation for Optimization

This phase involves implementation of our technique to optimize ACE. For this purpose, weights are added to all the terminals of ACE. There are usually two types of terminals present in the ACE, attribute values [A_1,…..A_n] and ephemeral constants [0-10]. Unique weights are assigned to all the terminals present in ACE by addition of a '*' node and a weight node. Let A be an ACE and t be the number of terminals in the classifier then the weight vector will be:-

$$[W_j] \text{ where } j=1:t \qquad 4\text{-}3$$

This process increases the complexity of the ACE and increases its depth by '1'. If the number of terminals present in the tree is equal to 't' then the increase in number of nodes in tuned tree is *2't'* where t nodes are function nodes having value '*' and 't' nodes are terminal nodes having weights as their values. This method scales the input of each terminal according to its weight. Let old terminal be T_o and new terminal be T_n, then the value of new terminal would be interpreted as

$$T_{nj}=W_j*T_{oj} \text{ where } j=1:t \qquad 4\text{-}4$$

An important point to note here is that the ACE remains intact if the values of all the added weights are set to '1'. Let A_0 be the original ACE and A_n be the weight added ACE, then

$$A_0 = A_n \; if \; \forall[W_j]=1 \; where \; j=1:t \qquad 4\text{-}5$$

Figure 4-2 illustrates an example, the ACE (A_1 / A_3) + 3 becomes (A_1 * W_1)/(A_3 * W_2) +(3 * W_3) after addition of weights. The added weights combine to form a weight vector = [W_1, W_2, W_3].

After addition of weights and defining a weight vector, this weight vector is optimized using PSO.

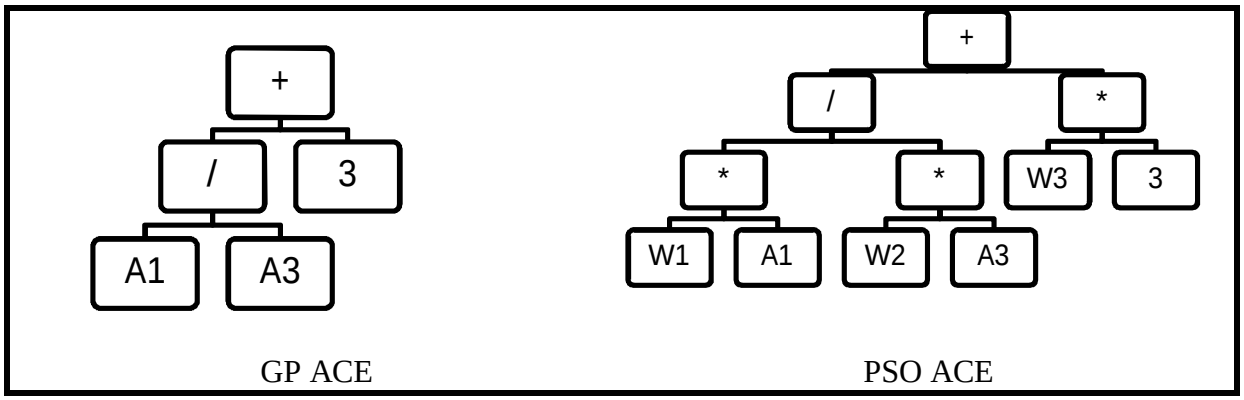

Figure 4-2 Addition of weights to classifiers for optimization

4.3.3 PSO Phase

The weight vector from equation 4-3 forms a multidimensional particle in PSO. The first and the most important step in PSO algorithm is initialization of position and velocities of particles. The empirical analysis of our problem revealed that random weight initialization does not lead to optimal results all the times. Whereas, one is interested in increasing the efficiency of classifiers or keeping it unchanged in the worst case. Therefore a special initialization mechanism has been incorporated that favors better particles. It has been seen that efficient initialization leads towards better solutions and tends to find the optimum solutions with less effort.

Typical random weight initialization has been coupled with a fitness checking measure. The fitness of each randomly initialized particle $Fit_{p,}$is calculated. A particle with fitness less than the original classifier Fit_{prev}, is not accepted. This random initialization process is repeated at most ten times (P_{count}) if solution of desired fitness is not found. After that, the best of ten is returned as the swarm particle. The iteration process is stopped if solution with better or equal fitness is found. Figure 4.3 presents the initialization algorithm used for PSO. The velocities of particles are initialized randomly. After the initialization, the PSO algorithm is continued in the traditional way. These randomly initialized fitter particles are then moved through the solution hyperspace in the search of better and optimal values. The movement of the swarm particles is controlled by their velocity and position update equations. The full model of PSO has been used with social and cognitive components for evolution

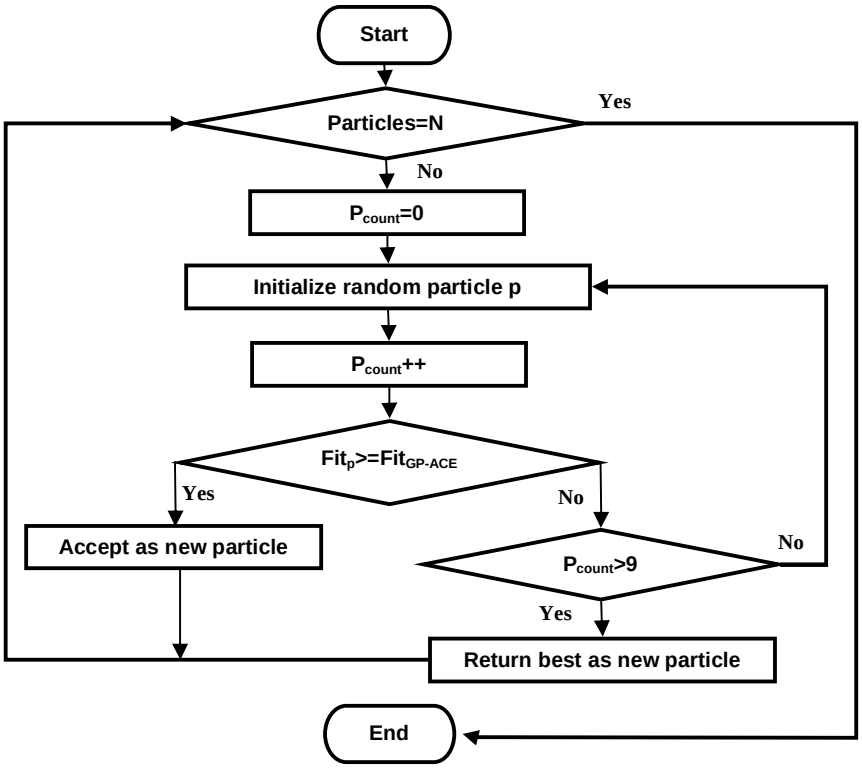

Figure 4-3 Initialization of weight population for PSO optimization, Fit_p= Fitness of a particle P, Fit_{GP_ACE}=Fitness of original classifier, P_{count}= Repetitions allowed

The velocity of particles is updated by the following equation:

$$V_{i+1} = \omega V_i + C_0 rand\ (0,1)(X_{pbest} - X_i) + C_1 rand\ (0,1)(X_{gbest} - X_i) \quad 4\text{-}6$$

Where *w* is the inertia weight, C_0 and C_1 are the constriction coefficients, X_{pbest} and X_{gbest} are the personal and global bests of the particle. The inertia weight can be interpreted as the fluidity of the medium in which a particle moves. Where as the constriction coefficients control the particle convergence and prevents explosion by eliminating the need for V_{max} parameter [100]. The Clerc's constriction method has been used which recomends following values

$$C_0=1.49618,\ C_1=1.49618 \text{ and } w=0.7298 \quad 4\text{-}7$$

These particles have ability to converge without any particular V_{max}. This results in a PSO with no problem specific parameters making it the canonical PSO of today as reported by Poli.[100] X_{lbest} is the best position a particle has ever visited, whereas the X_{gbest} is the best position of whole swarm. X_{gbest} can be different for each particle based on the model used for PSO topology.

The neighborhood model has been used where a particle follows its neighboring best position as opposed to global best particle where all particles in the swarm follow same particle. This helps in exploring the search space effectively. This has been suggested by Kennedy[92] that the neighborhood of smaller size works better for complex problems.

After velocity update, the particles update their positions using the following equation:

$$X_{i+1} = X_i + V_i \qquad 4\text{-}8$$

```
Step 1. Begin
Step 2. Initialize particles using algorithm in Figure 4.3.
Step 3. While termination criteria not met
     a. Calculate fitness of particles
     b. For each particle
               i.  Update particle g_best
              ii.  Update particle velocity using equation 4.4 and values
                   from 4.5
             iii.  Update Particle Position using equation 4.6
     c. End particles
Step 4. End while
Step 5. Return best particle
Step 6. End
```

Algorithm 4-2 PSO for Optimization

All the particles in the swarm continue to update their positions and velocities using above equations, moving in the multidimensional hyperspace of solutions in search of optimal position until the termination condition is met. This condition is either a number of iterations, or certain threshold of fitness achieved by best particle. The optimization algorithm is given in Algorithm 4-2. This is last of the three phases that form the proposed hybrid combination of GP and PSO (GPSO) technique.

Next Section provides an analysis of results obtained using GPSO for twelve benchmark problems.

4.4 Results

4.4.1 Experimental Setup

The datasets used for this experimentation are same as those of the previous chapter. We have compared the performance of the proposed GPSO with the DepthLimited GP proposed in the previous chapter.

Table 4.1 presents the PSO parameters used for GP classifier tuning in GPSO approach. Neighborhood PSO model has been used. Other parameters like constriction coefficients and inertia weight have been adopted from Clerc's analysis [89].These parameters have been selected after careful empirical analysis of the problem and relevant literature survey mentioned with the corresponding values of the parameters.

The GP results have been generated by application of tenfold cross validation twice each using different initial population. This process is repeated five times, each with a different tenfold partitioning and random sampling of the data. Ten GP runs are performed each involving tenfold cross validation. All the parameters in GP are kept same except the new initialization in each run. This helps to compare our results with other GP based classification techniques [54, 63].

Table 4.1 PSO parameters

Particles	100
Initial values	[-1,1]
Dimensions	Number of leaves
Iterations	30
C_0	1.49 [89]
C_1	1.49 [89]
W	0.7 [89]
Model	Lbest model [92]
Neighborhood size	2 [92]

Optimal values of constriction coefficients and inertia weights proposed by Clerc reported by Poli in [100] are 0.7298 and $C_0=C_1=1.49618$. In every single GP execution (from the total of 100), the best classifier is extracted after every 10 generations and recorded the classification

result for GP, having done that, PSO optimization has been performed ten times and recorded the average result for every classifier. These results are then averaged for all the GP executions and PSO executions.

4.4.2 Comparison with GP

This section compares the performance of GPSO classifiers with the classifiers obtained after GP evolution. The tables presented below show GPSO and traditional GP classification results in training as well as testing process for various datasets. A classifier is extracted after every 10 generations and analyzed the effect of optimization over various stages of evolution. As mentioned in previous section, all the presented results are averaged. The word average will not be used to avoid redundancy in the result discussion.

In case of HABER GPSO successfully increase the performance of GP classifiers from 76.4% to 77% for ten generations in training phase. In testing the performance of the classifier after ten generations was increased from 73.2% to 74.8%. For classifiers after 120 generations, the performance was increased from 72.5% to 74.4% using GPSO.

For WBC data highest accuracy of 97.7% has been achieved after completion of 120 generations at the expense of $7.2 * 10^4$ function evaluations. On the other hand, GPSO achieved better results after optimizing classifier obtained after 20 generations and achieved accuracy of 97.9%.The optimization of classifier achieved after 120 generations results in 99.6% accuracy. For test data the classifiers achieved 99.1% results.

For PARK data set the training accuracy increased from 80.6% to 83% yielding an increase of 2.4% for classifiers evolved for ten generations only. This increase was 0.7% for classifiers after 120 generations. For testing data the accuracy was increased to 2% after 10 generation classifier and 3.2% increase from 84.3% to 87.1% accuracy for classifier of 120 generations.

Table 4.2 Comparison of GP and GPSO for HABER

HABER										
	Training				Testing				GP NFE * 10^4	GPSO NFE * 10^4
Gen	GP%	S.D	GPSO%	S.D	GP%	S.D	GPSO%	S.D		
10	76.4	0.15	77.0	0.99	73.2	0.74	74.8	1.1	0.6	0.90
20	76.9	0.15	77.3	1.29	72.9	1.23	74.4	1.1	1.2	1.52
30	77.2	0.05	77.4	1.37	72.3	0.98	73.4	1.1	1.8	2.13
40	77.4	0.10	77.5	1.46	72.5	0.26	74.1	1.3	2.4	2.76
50	77.5	0.2	77.7	1.42	71.8	0.25	74.1	1.2	3.0	3.37
60	77.7	0.18	77.8	1.24	72.2	0.24	73.6	1.1	3.6	3.96
70	77.7	0.18	77.9	1.20	72.0	0.48	74.1	1.1	4.2	4.58
80	77.8	0.23	77.9	1.25	71.5	0.72	73.6	1.1	4.8	5.12
90	77.9	0.26	78.0	1.25	71.6	0.01	73.4	1.3	5.4	5.73
100	78.1	0.15	78.1	1.45	72.5	1.21	74.4	1.1	6.0	6.33
110	78.1	0.15	78.1	1.45	72.5	1.21	74.1	1.1	6.6	6.95
120	78.1	0.15	78.1	1.45	72.5	1.21	74.4	1.1	7.2	7.56

Table 4.3 Comparison of GP and GPSO for WBC

WBC										
	Training				Testing				GP NFE * 10^4	GPSO NFE * 10^4
Gen	GP%	S.D	GPSO%	S.D	GP%	S.D	GPSO%	S.D		
10	96.1	0.06	97.2	1.00	95.6	0.80	97.2	1.82	0.6	0.91
20	97.0	0.11	97.9	0.21	96.1	1.16	97.3	1.14	1.2	1.53
30	97.4	0.04	98.1	0.37	96.1	0.09	97.5	1.20	1.8	2.14
40	97.5	0.03	98.7	0.27	96.2	1.13	97.6	0.39	2.4	2.77
50	97.6	0.02	99.0	0.43	96.3	1.12	97.6	0.70	3.0	3.38
60	97.6	0.02	99.0	0.42	96.3	2.01	98.0	0.52	3.6	3.97
70	97.6	0.02	99.1	0.37	96.3	0.97	98.1	0.42	4.2	4.54
80	97.7	0.05	99.1	0.27	96.3	1.30	98.1	0.14	4.8	5.13
90	97.7	0.06	99.1	0.21	96.5	0.95	98.3	0.34	5.4	5.75
100	97.7	0.07	99.1	0.21	96.5	0.91	98.4	0.13	6.0	6.35
110	97.7	0.09	99.1	0.27	96.6	0.87	98.7	0.17	6.6	6.91
120	97.7	0.07	99.6	0.32	96.6	1.07	99.1	0.12	7.2	7.54

Table 4.4 Comparison of GP and GPSO for PARK

PARK										
	Training				Testing				GP NFE * 10^4	GPSO NFE * 10^4
Gen	GP%	S.D	GPSO%	S.D	GP%	S.D	GPSO%	S.D		
10	80.6	0.96	83.0	1.02	82.8	1.53	84.8	1.79	0.6	0.96
20	82.5	1.40	84.7	1.09	83.3	1.64	85.3	1.48	1.2	1.55
30	83.6	1.76	85.5	1.00	83.8	4.78	85.8	0.89	1.8	2.17
40	84.4	1.40	86.0	1.05	83.9	5.98	85.5	0.79	2.4	2.80
50	84.8	1.40	85.8	0.92	84.0	2.03	85.5	1.48	3.0	3.40
60	85.1	0.84	86.3	0.92	84.2	0.37	85.9	1.88	3.6	3.96
70	85.5	1.12	86.2	0.84	84.3	1.55	85.7	1.78	4.2	4.56
80	85.8	1.24	86.9	0.85	84.6	1.16	85.7	1.88	4.8	5.19
90	86.0	1.28	87.0	0.74	84.8	0.83	86.1	1.38	5.4	5.77
100	86.3	1.12	87.1	0.74	84.7	0.35	86.6	1.38	6.0	6.38
110	86.5	0.92	87.2	0.84	84.4	1.62	86.9	1.18	6.6	6.96
120	86.6	0.80	87.2	0.80	84.3	1.62	87.1	1.28	7.2	7.51

Table 4.5 Comparison of GP and GPSO for BUPA

BUPA										
	Training				Testing				GP NFE * 10^4	GPSO NFE * 10^4
Gen	GP%	S.D	GPSO%	S.D	GP%	S.D	GPSO%	S.D		
10	69.0	0.00	80.7	2.01	64.7	1.72	68.7	1.77	0.6	0.91
20	71.0	0.09	82.6	1.17	66.1	2.77	69.0	1.82	1.2	1.57
30	71.6	0.43	82.3	1.39	68.5	1.04	69.6	1.61	1.8	2.17
40	72.4	0.68	83.4	1.05	69.3	0.88	70.7	1.13	2.4	2.74
50	73.0	1.02	84.0	1.29	68.4	0.82	70.9	1.77	3.0	3.39
60	73.4	1.06	83.8	1.07	69.0	0.88	70.0	1.40	3.6	3.92
70	73.6	0.82	84.9	1.07	68.2	0.24	70.5	1.50	4.2	4.53
80	73.8	0.84	86.4	0.97	68.5	0.68	70.4	1.29	4.8	5.17
90	74.1	0.82	86.3	0.99	68.7	0.89	70.3	1.34	5.4	5.71
100	74.3	0.68	86.6	1.06	69.1	0.18	71.0	1.50	6.0	6.33
110	74.5	0.66	88.6	1.12	69.3	0.03	71.1	1.50	6.6	6.92
120	74.6	0.54	89.4	1.16	69.2	0.46	70.9	1.45	7.2	7.59

Table 4.6 Comparison of GP and GPSO for PIMA

PIMA										
	Training				Testing				GP NFE * 10^4	GPSO NFE * 10^4
Gen	GP%	S.D	GPSO%	S.D	GP%	S.D	GPSO%	S.D		
10	67.3	1.13	68.4	1.21	65.7	0.18	71.3	0.66	0.6	0.96
20	68.6	1.33	69.3	0.53	65.4	1.41	71.6	0.75	1.2	1.53
30	69.4	0.96	70.1	0.38	65.3	1.69	70.9	1.22	1.8	2.11
40	69.9	0.64	70.7	0.33	66.8	0.66	71.5	2.55	2.4	2.78
50	70.5	0.66	71.6	0.40	66.9	0.19	70.8	1.22	3.0	3.35
60	70.9	0.37	72.0	0.17	66.1	0.19	70.7	0.47	3.6	3.91
70	71.1	0.26	72.6	0.05	67.7	1.13	73.6	0.28	4.2	4.57
80	71.4	0.2	72.4	0.01	67.8	1.04	73.6	1.7	4.8	5.15
90	71.7	0.23	72.7	0.11	68.2	0.75	74.0	1.04	5.4	5.74
100	71.8	0.26	73.0	0.02	68.3	1.32	74.1	1.03	6.0	6.32
110	72.0	0.44	73.4	0.05	68.5	0.85	74.7	1.98	6.6	7.00
120	72.2	0.62	73.5	0.11	68.6	0.1	74.4	1.33	7.2	7.57

Table 4.7 Comparison of GP and GPSO for TRANS

TRANS										
	Training				Testing				GP NFE * 10^4	GPSO NFE * 10^4
Gen	GP%	S.D	GPSO%	S.D	GP%	S.D	GPSO%	S.D		
10	76.9	0.41	77.5	0.29	75.8	0.88	76.6	0.9	0.6	0.98
20	77.4	0.54	77.9	1.36	76.4	0.74	77.4	0.71	1.2	1.59
30	77.8	0.44	78.2	0.97	76.7	0.70	77.8	0.61	1.8	2.14
40	78.0	0.62	78.3	0.39	76.7	0.74	77.6	0.71	2.4	2.71
50	78.2	0.71	78.5	0.58	77.1	0.67	78.0	0.55	3.0	3.35
60	78.2	0.78	78.6	0.77	77.1	0.74	77.8	0.61	3.6	3.96
70	78.3	0.76	78.5	0.68	77.3	0.77	77.7	0.61	4.2	4.57
80	78.3	0.81	78.6	0.48	77.2	0.73	77.8	0.61	4.8	5.16
90	78.3	0.78	78.6	0.29	77.3	0.72	77.6	0.68	5.4	5.76
100	78.4	0.79	78.7	0.58	77.2	0.65	77.8	0.61	6.0	6.38
110	78.4	0.80	78.7	0.48	77.2	0.69	77.5	0.61	6.6	6.99
120	78.4	0.86	78.7	0.77	77.4	0.66	78.1	0.61	7.2	7.57

Table 4.8 Comparison of GP and GPSO for ION

ION										
	Training				Testing				GP NFE * 10^4	GPSO NFE * 10^4
Gen	GP%	S.D	GPSO%	S.D	GP%	S.D	GPSO%	S.D		
10	82.9	4.32	85.3	2.91	78.5	3.74	82.1	3.61	0.6	0.99
20	85.5	3.76	87.3	2.57	83.2	1.66	84.1	3.34	1.2	1.59
30	87.3	3.22	88.7	2.48	84.8	0.42	86.4	2.64	1.8	2.14
40	89.2	2.71	89.5	2.06	85.6	2.51	87.0	2.22	2.4	2.72
50	90.0	2.48	90.3	1.76	86.1	4.38	87.4	1.95	3.0	3.35
60	90.5	2.26	90.6	1.67	86.3	3.14	87.6	1.81	3.6	3.96
70	90.9	2.15	91.2	1.52	86.6	5.22	88.0	1.53	4.2	4.60
80	91.4	2.08	91.5	1.36	86.9	3.98	88.4	1.94	4.8	5.11
90	91.7	2.04	91.8	1.39	87.2	3.55	88.8	1.94	5.4	5.75
100	91.9	1.90	92.0	1.40	87.6	2.51	89.0	2.36	6.0	6.38
110	92.2	1.79	92.3	1.30	88.4	0.19	89.1	2.77	6.6	7.00
120	92.4	1.83	92.4	1.24	88.5	0.85	89.3	2.22	7.2	7.57

Table 4.9 Comparison of GP and GPSO for SPEC

SPEC										
	Training				Testing				GP NFE * 10^4	GPSO NFE * 10^4
Gen	GP%	S.D	GPSO%	S.D	GP%	S.D	GPSO%	S.D		
10	79.8	0.26	81.6	0.35	79.4	0.28	80.9	0.28	0.6	0.98
20	80.2	0.67	81.9	0.38	79.0	1.14	79.7	0.28	1.2	1.58
30	80.4	0.66	81.3	0.44	79.1	1.44	80.1	0.57	1.8	2.15
40	80.8	0.92	81.7	0.56	78.6	1.70	79.6	0.86	2.4	2.71
50	81.0	1.10	81.9	0.73	79.1	1.42	80.1	0.85	3.0	3.33
60	81.2	1.13	81.7	0.59	78.9	1.70	80.1	0.57	3.6	3.95
70	81.4	1.13	81.8	0.68	78.9	1.84	80.2	0.28	4.2	4.56
80	81.5	1.13	82.0	0.79	79.0	1.70	79.5	0.57	4.8	5.19
90	81.6	1.25	82.2	0.97	78.6	1.84	79.0	0.85	5.4	5.76
100	81.7	1.28	82.2	1.03	78.8	2.00	79.3	0.85	6.0	6.39
110	81.8	1.31	82.5	1.06	78.8	1.71	80.1	0.85	6.6	6.99
120	81.9	1.38	82.3	1.09	77.6	2.27	78.5	0.85	7.2	7.59

Table 4.10 Comparison of GP and GPSO for RIPPER

RIPPER										
	Training				Testing				GP NFE * 10^4	GPSO NFE * 10^4
Gen	GP%	S.D	GPSO%	S.D	GP%	S.D	GPSO%	S.D		
10	87.8	0.01	87.8	0.05	87.4	0.00	87.5	0.04	0.6	0.99
20	88.2	0.33	88.4	0.12	87.5	0.57	88.0	0.12	1.2	1.59
30	88.8	0.08	89.1	0.22	88.3	0.28	88.6	0.28	1.8	2.15
40	89.1	0.07	89.3	0.23	88.7	0.28	89.1	0.27	2.4	2.72
50	89.2	0.11	89.4	0.24	88.5	0.22	88.8	0.25	3.0	3.34
60	89.4	0.14	89.5	0.25	88.6	0.68	89.6	0.29	3.6	3.96
70	89.6	0.35	89.7	0.23	88.7	0.85	89.3	0.26	4.2	4.58
80	89.6	0.35	89.7	0.23	88.8	0.85	89.2	0.26	4.8	5.15
90	89.6	0.36	89.8	0.22	88.6	0.68	89.1	0.25	5.4	5.76
100	89.6	0.38	89.8	0.23	88.9	0.63	89.5	0.24	6.0	6.39
110	89.7	0.43	89.8	0.21	89.1	0.63	89.5	0.21	6.6	7.00
120	89.8	0.45	89.8	0.19	89.1	0.74	89.5	0.22	7.2	7.58

Table 4.11 Comparison of GP and GPSO for SONAR

SONAR										
	Training				Testing				GP NFE * 10^4	GPSO NFE * 10^4
Gen	GP%	S.D	GPSO%	S.D	GP%	S.D	GPSO%	S.D		
10	74.4	2.06	74.9	1.29	70.4	6.20	70.4	6.03	0.6	0.98
20	75.6	0.66	76.2	0.61	73.8	4.65	74.0	4.39	1.2	1.59
30	76.4	0.29	76.7	0.62	72.6	4.76	72.7	4.31	1.8	2.14
40	76.8	0.75	77.2	1.11	73.6	4.65	74.1	3.81	2.4	2.71
50	77.2	1.38	77.6	1.76	72.7	4.76	73.4	3.83	3.0	3.34
60	77.7	1.96	77.9	2.26	73.5	4.14	74.1	3.38	3.6	3.96
70	78.0	2.33	78.1	2.91	73.6	4.34	74.1	3.46	4.2	4.56
80	78.2	2.67	78.5	3.39	73.2	4.96	74.0	3.77	4.8	5.18
90	78.5	2.94	78.8	3.47	72.7	4.65	73.6	3.50	5.4	5.77
100	78.6	3.17	79.0	3.54	73.6	4.14	74.4	3.21	6.0	6.39
110	78.8	3.44	78.9	3.72	73.5	4.34	74.0	3.55	6.6	6.99
120	79.0	1.78	79.3	1.85	73.3	2.22	73.9	1.83	7.2	7.58

Table 4.12 Comparison of GP and GPSO for MUSK

MUSK										
	Training				Testing				GP NFE * 10^4	GPSO NFE * 10^4
Gen	GP%	S.D	GPSO%	S.D	GP%	S.D	GPSO%	S.D		
10	67.5	0.78	71.1	0.42	62.3	5.81	65.1	4.43	0.6	0.95
20	69.8	1.36	73.3	1.82	64.0	5.08	67.9	5.45	1.2	1.59
30	70.5	0.7	73.4	1.41	64.3	5.43	68.3	5.39	1.8	2.12
40	70.9	0.29	74.1	0.60	65.6	5.05	69.2	5.05	2.4	2.79
50	72.0	0.17	75.7	0.00	66.5	4.57	69.6	4.66	3.0	3.38
60	72.6	0.37	75.5	2.04	66.8	4.26	70.0	4.11	3.6	3.95
70	72.9	0.33	76.2	0.94	67.3	5.01	70.3	4.41	4.2	4.60
80	73.2	0.54	77.2	0.94	67.6	4.64	70.5	4.46	4.8	5.14
90	73.5	0.5	77.6	0.27	67.9	4.27	70.8	4.62	5.4	5.70
100	73.8	0.25	76.9	0.02	68.2	4.61	71.2	4.55	6.0	6.39
110	74.1	0.45	77.0	1.55	68.5	4.25	72.1	4.96	6.6	6.91
120	74.3	0.54	77.2	0.09	69.3	3.83	73.4	3.58	7.2	7.51

Table 4.13 Comparison of GP and GPSO for HRT

HRT										
	Training				Testing				GP NFE * 10^4	GPSO NFE * 10^4
Gen	GP%	S.D	GPSO%	S.D	GP%	S.D	GPSO%	S.D		
10	77.5	0.43	79.6	3.15	73.38	4.24	75.6	4.17	0.6	0.94
20	79.3	1.31	82.17	3.29	77.5	0.05	79.6	3.72	1.2	1.53
30	80.3	0.73	82.79	2.66	78.3	6.85	81.6	5.08	1.8	2.10
40	81.9	0.58	84.16	1.89	80.4	0.05	82.7	2.32	2.4	2.74
50	82.7	0.58	85.61	2.13	82.3	2.94	84.0	3.28	3.0	3.40
60	83.2	1.01	85.90	4.21	82.31	7.02	84.7	2.72	3.6	3.93
70	83.3	1.16	86.77	4.07	83.35	5.55	85.4	1.36	4.2	4.54
80	83.8	0.73	87.19	3.34	88.27	5.49	88.3	1.41	4.8	5.10
90	83.8	0.73	87.63	3.34	88.27	5.49	90.7	1.41	5.4	5.72
100	84.1	1.16	86.94	3.77	88.27	5.49	90.8	1.41	6.0	6.32
110	84.1	1.16	86.69	3.77	88.27	5.49	91.1	1.41	6.6	6.94
120	84.1	1.16	88.09	3.77	88.27	5.49	91.7	1.41	7.2	7.53

In case of BUPA the maximum performance achieved by GP is 74.6% after 120 generations and GPSO outperformed it by optimizing the classifiers obtained after 10 generations with 80.7% accuracy. This higher performance has been achieved with only 1.2 * 10^4 NFE for GPSO as compared to 7.2 * 10^4 NFE for GP. GPSO achieved the accuracy of 89.4% for optimizing classifiers obtained after 120 generations. The testing accuracy was 69.2% and 70.9% for GP and GPSO respectively. For PIMA after 120 generations, the GP accuracy was 72.2% that was increased to 73.5% by GPSO. The testing accuracies were 68.6% and 74.4% respectively. For TRANS dataset GPSO has increased the accuracy from 78.4% to 78.7% and 77.5% to 78.1% in training and testing phases. GPSO successfully increased the performance for ION data. In testing, the accuracy was increased from 88.5% to 89.3%. GPSO increased the accuracy of SPEC data from 81.9% to 82.3% and 77.6% to 77.5% for training and testing data for classifiers of 120 generations. The accuracy for sonar data has been increased from 73.3% to 73.9% for testing phase for classifiers of 120 generations. GPSO successfully increased the accuracy of MUSK data from 74.3% to 77.2% in training, or testing phase the accuracy increase from 69.3% to 73.4% for the classifiers after 120 generations

For HRT data GPSO offered an increase of 3.9 and 3.43 for training and testing data. This can be noted here that in most of the cases GPSO outperforms GP both in terms of accuracy and efficiency i.e giving better results in much less number of function evaluations.

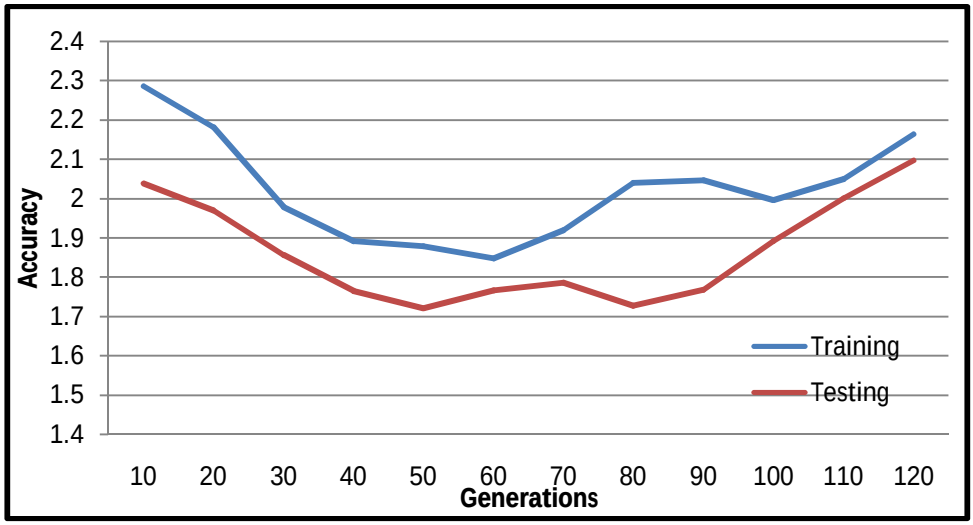

Figure 4-4 Average increase in accuracy (training and testing) for all datasets using GPSO

Figure 4.4 gives an overview of the performance of the proposed algorithm. It can be noticed that average performance of PSO is better for all the data sets. PSO phase has achieved average

accuracy increase of more than 1.7% in all the cases. This increase is higher in earlier generations, when the evolved structures are trying to mature. The increase is higher in later generations, when trees have been affected by bloat. This demonstrates that GPSO can increase efficiency of less efficient classifiers.

Table 4.14 Fitness per NFE for GP and GPSO

Average change in fitness per FE			
Dataset	**Mode**	**GP**	**PSO**
WBC	Train	0.004181	0.028869
	Test	0.004139	0.028653
BUPA	Train	0.003079	0.021343
	Test	0.002889	0.020454
HABER	Train	0.003323	0.022798
	Test	0.003130	0.021715
PARK	Train	0.003586	0.023525
	Test	0.003404	0.022625
PIMA	Train	0.002985	0.020635
	Test	0.002858	0.020904
TRANS	Train	0.003346	0.021661
	Test	0.003298	0.021453
ION	Train	0.003750	0.025007
	Test	0.003590	0.024187
SPEC	Train	0.003471	0.022557
	Test	0.003407	0.021966
RIPR	Train	0.003820	0.024415
	Test	0.003795	0.024315
SONAR	Train	0.003285	0.021348
	Test	0.003119	0.020200
MUSK	Train	0.003030	0.021635
	Test	0.002792	0.020043
HRT	Train	0.003466	0.025785
	Test	0.003409	0.025867

Table 4.14 displays the fitness per NFE for GP and PSO. It is clear that PSO phase results in higher fitness per NFE. This shows that PSO can offer better results in less function evaluations for optimizing a particular expression obtained using GP.

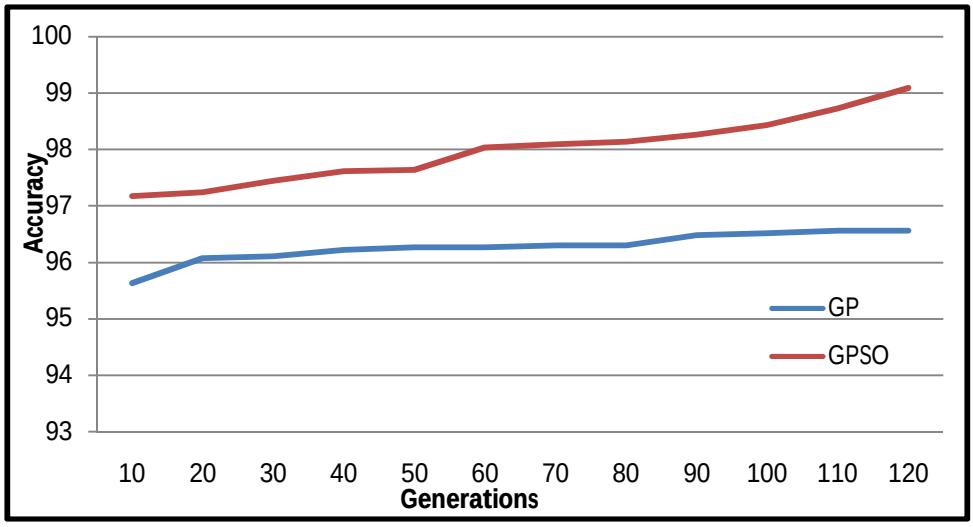

Figure 4-5 Increase in accuracy (test) after optimization for WBC

Figure 4-5 presents a graphical illustration of the increase in efficiency achieved after optimization for WBC data. This can be seen that the optimization process successfully achieves the optimal results that were not achieved by GP alone till the end of evolution with larger NFE.

Another factor that can be noted is the average increase in fitness for all the generations in training or testing data. This gain is mentioned in Table 4.15. It can be concluded that GPSO exhibits good generalizing abilities for almost all of the datasets.

Table 4.15 Average performance gain for all generations

Dataset	Train	Test
WBC	1.30	1.70
BUPA	11.9	2.00
HABER	0.16	1.73
PARK	1.20	1.80
PIMA	1.07	5.40
TRANS	0.35	0.69
ION	0.58	1.45
SPEC	0.81	0.94
RIPR	0.14	0.45
SONAR	0.32	0.51
MUSK	3.34	3.34
HRT	2.95	2.27

Table 4.16 presents some classifiers with their respective optimized versions. The classifiers are represented by their corresponding prefix notations. We can see that the addition of weights along all the terminals increase the depth of a classifier by one. But this increase adds to the classification accuracy of classifiers. We can see that the added weights provide a proportion to which a terminal participates in the final classification decision. This weight addition and tuning is a local search mechanism for efficient classifiers in the vicinity of a good structure.

Table 4.16 Original and optimized classifiers with their accuracies for PIMA

	Classifiers	Accuracy
GP	- A3 A6	67.0 %
GPSO	- * A3 0.51 * A6 0.30	74.1 %
GP	* + / A3 A1 * A2 A5 + - A3 A2 - A3 8	69.9 %
GPSO	* + / * A3 0.50 * A1 0.58 * * A2 0.75 * A5 0.56 + - * A3 0.29 * A2 0.49 - * A3 0.62 * 8 -0.79	71.8 %
GP	- + A4 * A2 / A2 + - A6 A1 A2 + A3 + A3 A8	65.0 %
GPSO	- + * A4 0.15 * * A2 0.99 / * A2 0.39 + - * A6 0.02 * A1 0.54 * A2 0.59 + * A3 0.68 + * A3 0.65 * A8 0.12	67.0 %
GP	- A1 / 5 A7	68.9 %
GPSO	- * A1 0.65 / 5 * 0.79 * A7 0.22	69.3 %
GP	- / A1 / 5 A7 / A2 A2	68.3 %
GPSO	- / * A1 0.13 / * 5 0.7 * A7 0.97 / * A2 -0.25 * A2 -0.27	74.6 %
GP	- * - A1 7 + A3 A1 / / A5 A4 - A5 A6	68.3 %
GPSO	- * - * A1 -0.1 -0.73 + * A3 -2.3 * A1 -2.3 / / * A5 0.06 * A4 -2.34 - * A5 -1.82 * A6 -0.47	69.3 %

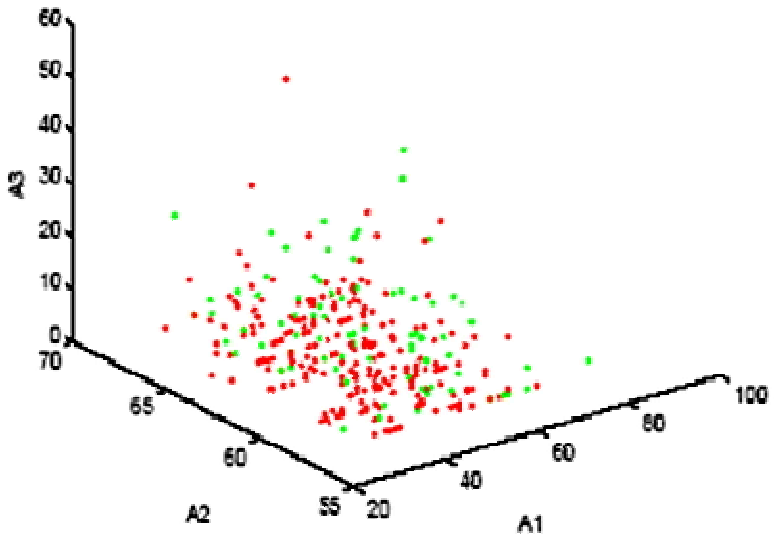

Figure 4-6 Plot of HABER Data

Figure 4-6 presents two classes of Haber dataset. This three dimensional data can be plotted to illustrate the classification decisions in our example. Figures 4-7 and 4-8 present the classifier decision before performing the tuning process. Both figures present two different views of same classifiers. This classifier has 79.3 % accuracy over the Haber test dataset.

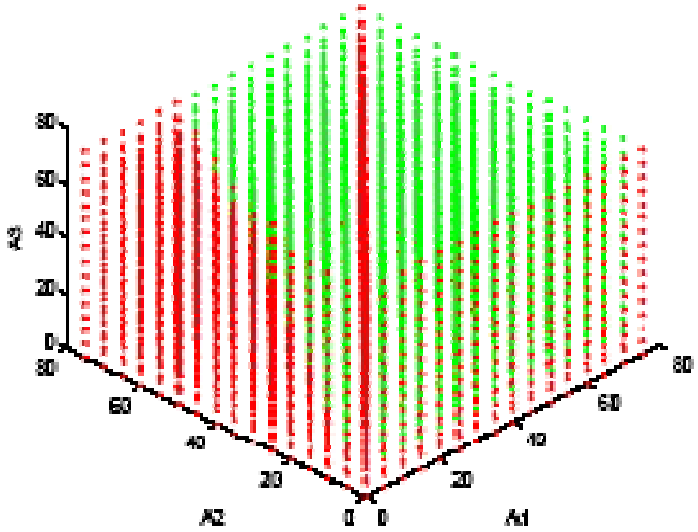

Figure 4-7 GP classifier for HABER data

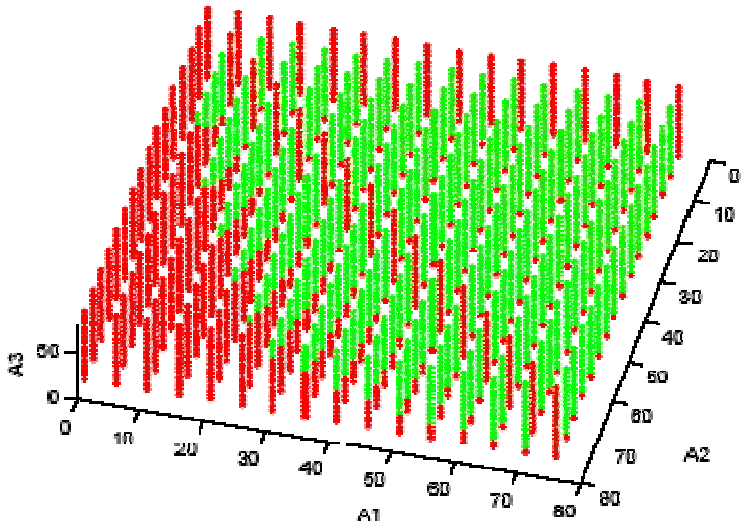

Figure 4-8 Rotated view of GP classifier for HABER data

The figures 4-9 and 4-10 present the classification decision of former classifier after performing the tuning process. We can see that the classification boundaries have been shifted as the result of tuning. This shift has increased the testing accuracy from 79.3% to 82.7 percent. These figures portray the behavior of proposed tuning phase and its potential for further enhancements.

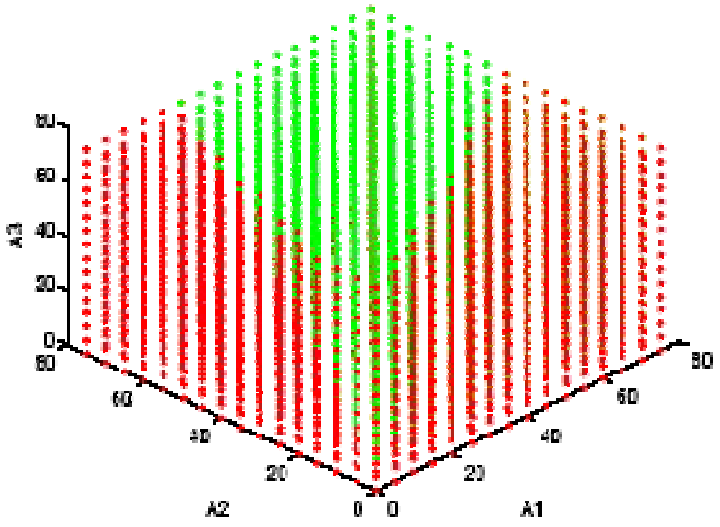

Figure 4-9 GPSO classifier for HABER data

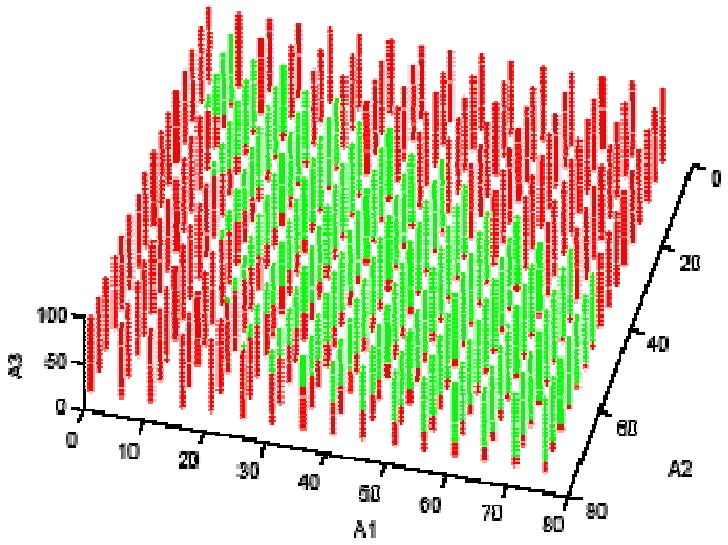

Figure 4-10 Rotated view of GPSO classifier for HABER data

4.4.3 Comparison with other Possibilities of Weight Addition

In this section the proposed optimization technique is compared with other possible weight addition and optimization scenarios. These possible methods are:-

1. Optimization of ephemeral constants by addition of weights GPSO-1

   In this method weights are added to ephemeral constant terminals only and these weights are optimized using PSO. Other terminals containing attribute values remain intact.

2. Optimization of constants by treating constants as weights GPSO-2

   The ephemeral constants present in expression are treated as weights and optimized using PSO.

3. Optimization of attribute weights only GPSO-3

   Weights are added associated to each attribute and the constants remain intact.

4. Optimization of attributes and ephemeral constants GPSO

   This is the proposed methodology where weights are added along each terminal and optimized using PSO.

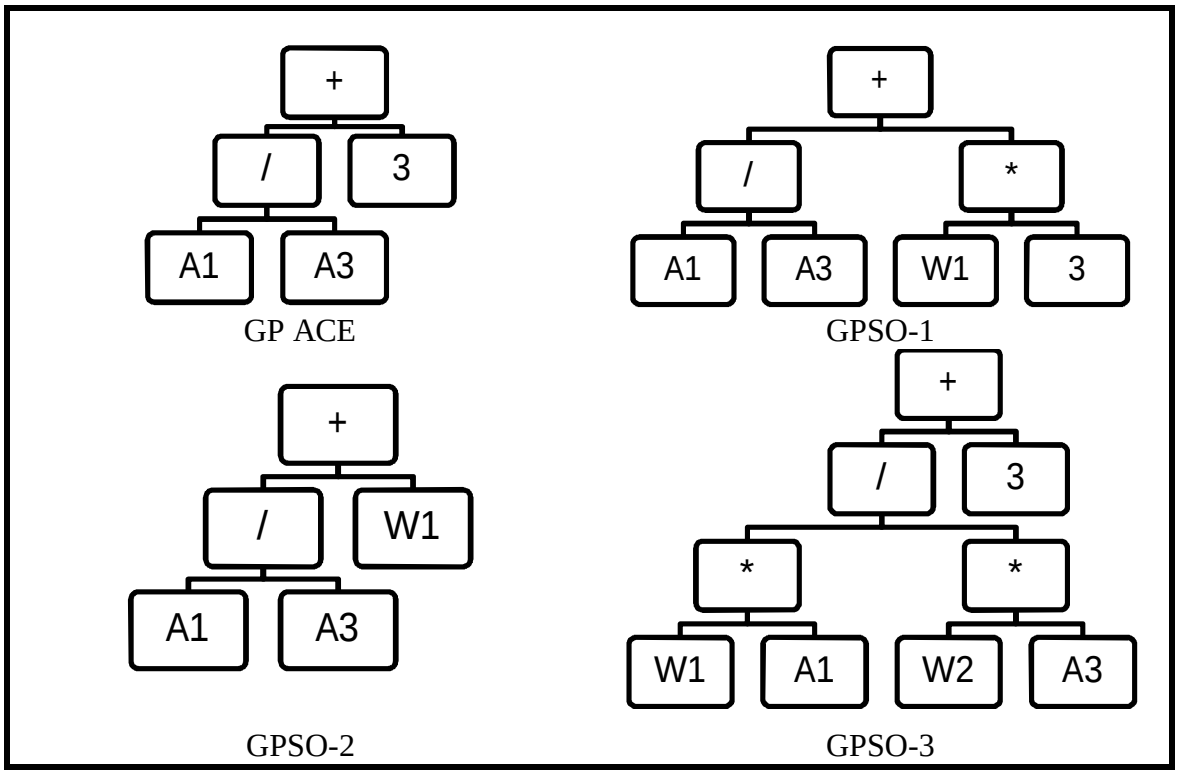

Figure 4-11GPSO Variants (A_1-A_n => Attributes of Data, W_1-W_n => Added Weights)

These variants are presented in Figure 4.11. The results of the new optimization techniques are presented in the following tables. Results present accuracy gain, averaged over all classifiers obtained after tenfold cross validation. The average gain in accuracy is around 2.7% per classifier for GPSO. This is higher as compared to other modes of optimization where gain remains less than 1 in most of the cases.

As mentioned in Table 4-17 to Table 4-21 GPSO-1 has increased the performance of classifiers up to some extent in case of BUPA, PIMA and WBC on the test data. On the other hand it has not been able to increase the performance of classifiers in case of PARK and HABER.

GPSO-2 and GPSO-3 has also succeeded in increasing the performance of classifiers demonstrating noticeable performance for GPSO-3. GPSO-2 failed to optimize classifiers for PARK data.

Table 4.17 Increase in accuracy (test) using GPSO-1

Generations	PARK	BUPA	PIMA	HABER	WBC
10	0.00	1.82	1.07	0.34	0.53
20	-1.70	0.61	0.67	0.00	-0.50
30	0.56	0.30	0.00	0.33	0.19
40	2.22	1.52	0.27	0.70	0.72
50	-1.70	-0.30	0.53	0.34	0.54
60	1.67	1.23	0.40	-0.70	-0.70
70	-1.10	0.31	1.20	-0.30	0.00
80	-0.60	1.22	0.67	0.00	-0.20
90	0.00	0.62	0.27	0.34	0.90
100	0.00	1.52	0.13	-1.00	0.36
110	-1.10	0.61	-0.30	-1.00	0.19
120	-1.10	1.52	-2.30	-0.30	0.54
Average Gain	-0.20	0.92	0.22	-0.10	0.21

Table 4.18 Increase in accuracy (test) using GPSO-2

Generations	PARK	BUPA	PIMA	HABER	WBC
10	-0.60	1.21	0.80	0.36	0.00
20	-1.10	0.61	0.40	0.70	-1.10
30	-1.70	-0.30	0.40	0.33	0.19
40	0.56	0.62	0.27	0.00	0.54
50	0.56	1.22	0.27	1.05	0.36
60	0.00	1.22	0.53	0.36	-0.70
70	-0.60	1.52	1.47	1.39	0.00
80	0.56	0.62	1.33	0.36	-0.20
90	-0.60	0.31	0.93	0.70	0.54
100	-1.10	0.92	0.67	0.01	0.36
110	-1.10	1.52	0.00	0.01	0.19
120	0.00	0.91	-0.30	1.05	0.54
Average Gain	-0.40	0.87	0.57	0.53	0.06

Table 4.19 Increase in accuracy (test) using GPSO-3

Generations	PARK	BUPA	PIMA	HABER	WBC
10	1.96	1.17	2.13	0.69	1.30
20	2.80	1.17	2.53	0.69	1.30
30	1.69	2.67	1.07	0.69	0.65
40	1.39	-0.10	1.07	1.91	1.08
50	2.24	0.49	2.26	1.75	0.95
60	2.24	2.32	2.14	2.26	1.52
70	1.69	1.00	1.74	1.75	1.11
80	1.98	0.32	1.34	1.92	1.91
90	1.10	1.34	2.01	1.91	0.06
100	0.85	0.17	3.61	1.06	0.34
110	1.96	2.12	1.20	1.23	0.30
120	1.69	0.36	1.47	1.39	1.23
Average Gain	1.80	1.09	1.88	1.43	0.98

Table 4.20 Increase in accuracy (test) using GPSO

Generations	PARK	BUPA	PIMA	HABER	WBC
10	3.8	4.0	5.6	1.6	1.6
20	2.2	2.9	6.2	1.5	1.2
30	3.2	1.1	5.6	1.1	1.4
40	3.1	1.4	4.7	1.6	1.4
50	1.5	2.5	3.9	2.3	1.3
60	2.9	1.0	4.6	1.4	1.7
70	2.9	2.3	5.9	2.1	1.8
80	1.7	1.9	5.8	2.1	1.8
90	1.7	1.6	5.8	1.8	1.8
100	2.7	1.9	5.8	1.9	1.9
110	2.7	1.8	6.2	1.6	2.1
120	4.6	1.9	5.8	1.8	2.5
Average Gain	2.75	2.03	5.49	1.73	1.71

Table 4.21 Average gain in test accuracy of different optimization methods

	PARK	BUPA	PIMA	HABER	WBC
GPSO	2.75	2.03	5.49	1.73	1.71
GPSO-1	-0.20	0.92	0.22	-0.10	0.21
GPSO-2	-0.40	0.87	0.57	0.53	0.06
GPSO-3	1.80	1.09	1.88	1.43	0.98

Table 4.21 shows the average performance gain for the four optimization methods. This can be seen that the GPSO has outperformed other optimization cases revealing its worth and effectiveness. This gain is superior to other optimization scenarios for all the datasets which have been used in the experimentation.

4.4.4 Comparison with other GP Based Techniques

Table 4.22 compares the performance of GP and GPSO with other classification techniques.

Table 4.22 Comparison with other GP based techniques

Datasets	GP	GPSO	Others
WBC	96.56%	99.09%	95.1%, MultiTree GP [54], 96.4%, Dynamic Range GP [50], 93.92-97.54%, Rule GP [32] , 97.9%, Enhanced GP [63], 98.2%, Simplification GP [66], 92.5-96.24% GP structures [24]
BUPA	69.2%	72.67%	68.4%, Dynamic Range GP [50], 60.8%, MultiTree GP [54]
PIMA	68.1%	74.1%	74.2%, Dynamic Range GP [50], 68.3-75.75% GP structures [24], 68.64-75.16% Rule GP [32] , 74.87-75.53% Enhanced GP [63]
ION	88.5	89.3	85.4-90.52% Rule induction [25]
SPEC	77.6	78.5	83.2±7.3, Simplification GP [66]
SONAR	73.3	73.9	68.4-72.42% Rule induction [25]
HRT	73.38	75.6	71.4-79.66% GP structures [24], 78.01-92.81% Rule GP [32] , 74.2-77.9% Rule induction [25]

This kind of comparison is usually not encouraged due to different partitions of data used for experimentation. To overcome this limitation, we have performed rigorous experiments using different partitions of the data. In total GP was evolved 100 times. And for each GP classifier PSO was evolved 10 times. So each entry in the table is an average of 10,000 executions of GPSO algorithm. Making it rigorous enough to be compared with the other reported results.

4.5 Conclusion

A novel hybrid method for efficient tuning of GP evolved ACE is presented. We have successfully proven that addition of optimal weights to the terminals of an ACE can increase its performance. This method can eliminate the need of evolving GP for larger number of generations, which is a complex task. An intelligent structure of ACE can be extracted in a few generations of GP and effectively increase its performance by addition and optimization of weights. The process performs better in smaller number of NFE. This method has performed equally well in other GP based classification approaches. The performance of presented method is compared with other weight addition methods. Addition of weights along all the terminals yields better results.

Chapter 5. Classification of Mixed Variable Data

5.1 Introduction

Most of the real world data contains some continuous variables like age, weight, temperature or pressure mixed with some variables like gender, color or nationality. The former variables are known as continuous /numerical or real valued where as the later ones are called categorical/nominal variables. This mixed type data is the common form of data encountered in most real life problems. On the other hand, most of the data analysis algorithms are applicable to only one data-type. All other data-types are usually converted or processed before use by such algorithms. This type of preprocessing / conversion adds to the computational effort and may result in loss of precious information and is a possible source of bias. GP based classification algorithms, in this context, behave similarly.

GP has been applied for classification in two different ways; one evolves classifiers as logical rules applicable to categorical data and the continuous attributes must be descritised; another method is evolution of arithmetic classifier expressions applicable to continuous attributes and the categorical attributes have to be encoded into numeric values. In this chapter, a method that eliminates the need of data transformation for classification is presented. The proposed two layered GP has produced compatible results as compared to other techniques presented in the literature.

5.2 Related Work

Classification for mixed data-type problems has received relatively less attention by researchers, despite the fact that most of the data in real life is of mixed type. Very few researchers have focused on classifying mixed type data, and GP based classification algorithms are no exception.

Some of the recent literature on the topic includes atomic representations for GP based classifiers, introduced by Eggermont [30]. 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. Loveard [70] proposed two explicit strategies to use nominal attributes for mixed type data classification, one method is conversion of a nominal attribute to binary or numeric values(numeric conversion). Numeric conversion assigns a numeric value starting from 1 up to maximum values attained by an attribute. Binary conversion assigns a true value for presence of an attribute and 0 for absence. Both these methods introduce new type of terminal nodes in the trees which can increase with the increase in number of categorical attributes and their possible values. These terminal types are different for each data set. Both numeric conversion methods require data conversion, increasing the computational overhead and possible source of bias. The second method considers splitting of GP execution based upon the value of a nominal attribute (execution branching). This is same as decision trees where one branch is taken based upon the value of the attribute. This method requires definition of new node types for every problem. For categorical attributes with lager possible values, the tree size will increase proportionally increasing the size of search space. Ong [101] in his credit approval application used the discretization of continuous attributes present in data.

Most of the above mentioned methods involve data conversions requiring a preprocessing step. This step increases the complexity and can be a possible source of a biasness and loss of information. 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’ or ‘strongly typed GP’. There is a type restriction on which terminal and function can be chosen for a particular node. Each child node must have a return type expected by its parents. Such types of constraints reduce the search space and can be a likely source of bias. Constrained syntax GP has been used in [33] and [102].

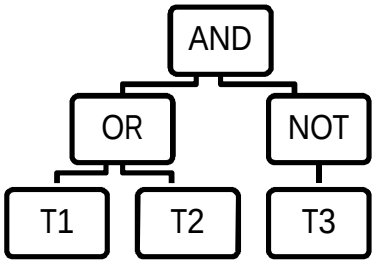

Figure 5-1 Outer Logical Tree

5.3 Proposed Two Layered Approach

The first step in creating a GP system is the solution representation with selection of function and terminal set. The proposed solution is a two layered tree. The outer layer is a logical layer with function and terminal nodes. The function set is given in Table 5.1 The terminal nodes of outer layer tree are the inner layer trees. An example representation of outer logical tree presented in Figure 5 2 is

```
Outer Logical Tree = (Inner tree1) AND (Inner tree2) OR
(Inner tree3)                                              5-1
```

These trees are of two types in our approach.

1) Logical inner tree
2) Arithmetic inner tree

Logical Inner Tree combines the categorical attributes present in the data with logical functions. The node in this tree is ‘categorical attribute 1=any value’ or ‘categorical attribute 1≠any value’. The functions in logical inner tree are the same as logical outer tree mentioned in Table 5.1. An example representation of inner logical tree presented in Figure 5-2 is:

```
Logical Inner Tree = (C1=’a’) AND (C3=’b’) OR (C4≠’c’)     5-2
```

The output of a logical tree is a boolean value for each data instance, indicating if the tree satisfies the given instance or not. The Arithmetic Inner Tree combines the continuous / numerical attributes of the data by arithmetic functions. The function set used by this representation is mentioned in Table 5.1. The terminals in this representation are the attribute names which are replaced by values of an instance for evaluation. An example representation of an arithmetic inner tree also presented in Figure 5-2 is:

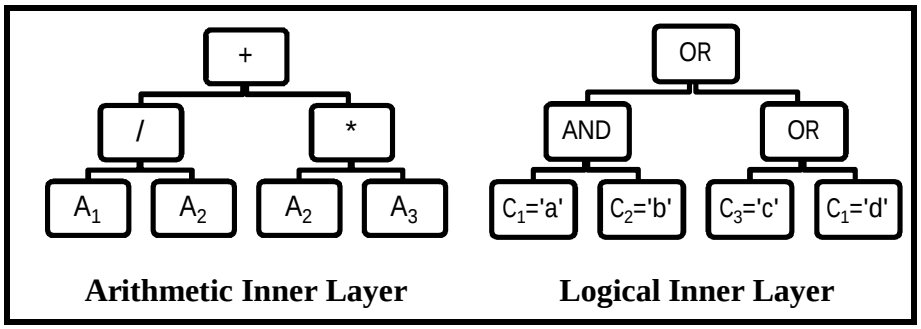

Figure 5-2 Inner layers of layered classifier A_1-A_n=> Numerical Attributes of Data, C_1-C_n => Categorical Attributes of Data

Arithmetic Inner Tree = (A1/A2) + (A3*A8) 5-3

The output of an Arithmetic tree is a real value depending on the value of each attribute of a specific instance. Several Arithmetic and Logical trees provide their output as input to outer logical tree. The output of a logical inner tree can be directly used by the outer logical tree. The real output of arithmetic inner tree is considered true for positive values and false for negative values.

Initialization of random population is crucial step for evolutionary algorithms. We investigated various different depths and concluded that ramped half and half initialization method with depth ranging from 2 to 4 for outer layer works better. The method allows diverse and flexible population initialization and has been found successful for variety of classification and other problems. The inner layer trees are also randomly created by ramped half and half method with depth limits 2-4. This implies that the maximum depth of a tree cannot exceed the level eight. Therefore, the maximum number of, possible antecedents in an expression are sixteen. Where, each of the antecedents, itself, can be an expression of no more than sixteen variables.

Three operators are used for the evolutionary process; reproduction, mutation and crossover. The reproduction operator simply transfers an individual to the next generation. The individual member is chosen using fitness proportionate selection. The individual for mutation is chosen randomly. The mutation operator randomly selects an inner subtree with 50% probability, if the root of subtree is a logical node, the subtree is replaced with a logical expression. On the other

hand if the root of the subtree is an arithmetic node, the subtree is replaced with a randomly generated arithmetic subtree. If the selected node is from the outer tree a random outer subtree is generated and replaces the original subtree.

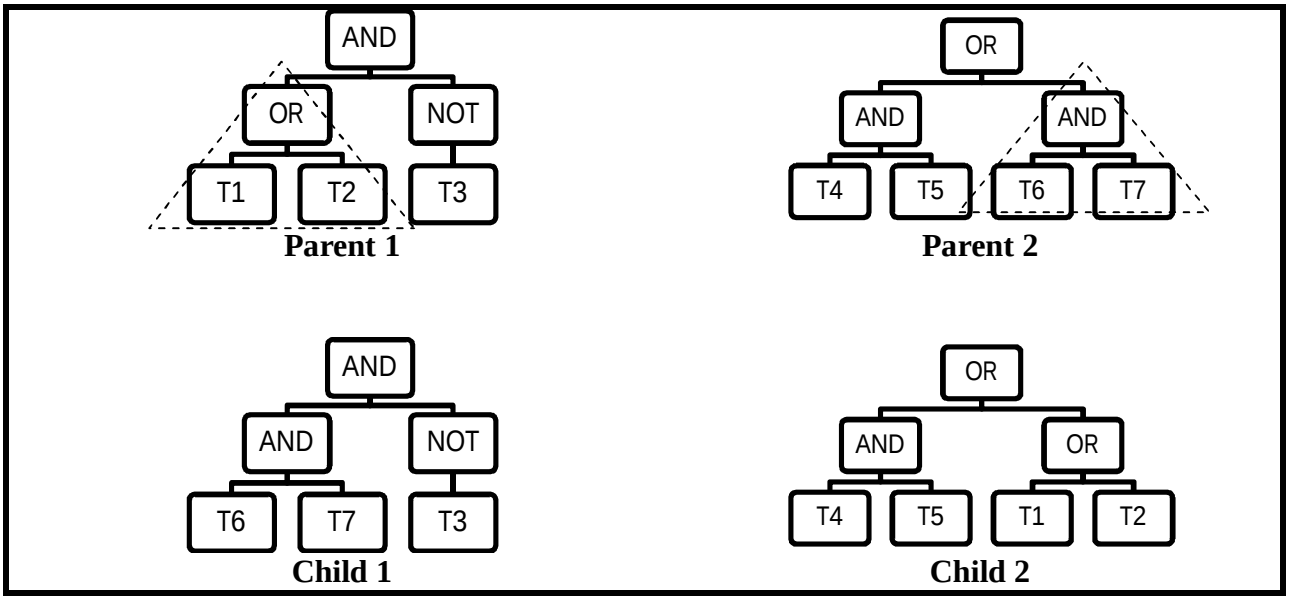

Figure 5-3 Crossover operator for layered classifier

For crossover a random node is selected from the outer tree of both parents and swapped. The only restriction placed on both subtrees (outer) is that they should have same depth. This restriction has been adopted from DepthLimited Crossover [65] where subtrees of same depth are swapped to prevent increase in classifier size during evolution. The process is illustrated in the Figure 5-3 where T1-T7 are the inner nodes of varying depths.

```
Step 1:  Begin
Step 2:  For each data instance
      a. Evaluate logical antecedents using attributes values of given
         Instance (Inner Layer)
      b. Evaluate arithmetic antecedents using attributes values of
         given Instance (Inner Layer)
      c. Evaluate the outer expression by substituting antecedent values
      d. If expression ==true and data instance ∈ class 1
      e. Increment CorrectCount
      f. If expression = false and data instance ∉ class 1
      g. Increment CorrectCount
Step 3:  End for
Step 4:  Fitness= CorrectCount /number of data instances
Step 5:  End
```

Algorithm 5-1 Fitness Accuracy (ACC) for Classification

```
Step 1:  Begin
Step 2:  For each data instance
      a. Evaluate logical antecedents using attributes values of given
         Instance (Inner Layer)
      b. Evaluate arithmetic antecedents using attributes values of
         given Instance (Inner Layer)
      c. Evaluate the outer expression by substituting antecedent
         values
      d. Count TruePositive, TrueNegative, FalsePositive and
         FalseNegative
Step 3:  End for
Step 4:  Fitness=  0.5  *[(  TruePositive/(  TruePositive  +
       FalseNegative))+( TrueNegative/( TrueNegative + FalsePositive))]
Step 5:  End
```

Algorithm 5-2 Fitness Area under the Convex Hull (AUC) for Classification

Two different fitness functions have been investigated in our approach. One is the classification accuracy for the classifier and the other is Area Under the Convex hull (AUC) or Area Under the Curve. This was done because the accuracy is not considered a valid measure in the case where data is largely skewed i.e. the number of instances belonging to one class are greater than the other. One the other hand a classifier with higher AUC is considered to have better discrimination ability. The results agree to the general observation. Both fitness functions are described in Algorithm 5-1 and Algorithm 5-2. The expression is evaluated for each data instance and accuracy is calculated. The overall classification algorithm is explained in Algorithm 5-3.

```
Step 1:  Begin
Step 2:  Set population size =N
Step 3:  Randomly initialize N GP expressions with logical function sets
       and antecedents   (outer layer)
Step 4:  For each antecedent in each GP expression initialize a random
       arithmetic or logical expression with equal probability.(Inner
       layer)
Step 5:  Calculate fitness of each tree using Algorithm fitness
Step 6:  Evolve GP expressions using operators
Step 7:  Return best GP expression
Step 8:  End
```

Algorithm 5-3 Layered GP (L2GP) for Classification

5.4 Results

Table 5.1 lists the initial parameters used in the proposed GP system. The reported experiments have been conducted by making five different tenfold random partitioning of each dataset. For each random partitioning, GP system has been evolved two times for tenfold cross validation. This was done to be consistent with other works reported in literature [49].

Table 5.1 GP Parameters used for layered classification

Initialization method	Ramped half and half depth 2-4
Outer Logical Layer function set	AND, OR, NOT
Arithmetic Inner Layer Function set	+ , * , -, / (protected division)
Inner logical Layer Function set	AND, OR, NOT
Inner maximum depth	4
Outer maximum depth	4
Termination criteria	User defined maximum generations
Arithmetic terminals	Numerical attributes, Constants[0,10]
Logical terminals	Categorical attribute values
Mutation probability	0.25
Reproduction Probability	0.25
Crossover Probability	0.5
Arithmetic node probability	0.5
Logical node probability	0.5
Mutation criteria	Random
Crossover selection	Tournament Selection with size 7
Reproduction criteria	Fitness proportionate selection

The datasets used in the empirical analysis have been taken from the UCI ML [84] repository. The properties of data sets are mentioned in Table 5.2. This investigation has been restricted to binary classification problems only.

Table 5.2 Datasets used in classification

Dataset	Instances	Attributes	Distribution
Japanese credit (CRD)	690	15, 6 continuous , 9 categorical	44.5, 55.5
Australian Credit (AUS)	690	14, 6 numerical , 8 categorical	44.5, 55.5
German Credit (GER)	1000	20, 7 numerical , 13 categorical	30, 70
Heart Disease (HRT)	270	13, 6 real , 7 binary, nominal, ordered	44.4, 55.6
Hepititus (HEP)	155	20, 5 numerical ,15 categorical	20.6, 79.4

Different measures have been used to examine the performance of our proposed approach. These measures are [103]:-

- **Accuracy**

  Accuracy is the prediction rate of a classifier and shows the percentage of the number of instances correctly classified.

$$Accuracy = \frac{tp+tn}{tp+tn+fp+fn} \quad 5\text{-}4$$

  Where *tp*=true positives, *tn*=true negatives, *fp*=false positives and *fn*=false negatives

- **Sensitivity**

  It can also be stated as the probability that a positive test, results as positive. The higher the sensitivity, the higher the positives remain correctly detected. It is also known as TPR.

$$Sensitivity = \frac{tp}{tp+tn} \quad 5\text{-}5$$

- **Specificity**

  It measures the proportion of negatives identified as negatives correctly.

$$Specificity = \frac{tn}{tn+fp} \quad 5\text{-}6$$

- **Area under the curve**

  Area under the curve (AUC) has been defined as a measure that correctly evaluates the discriminating power of a classifier and acts as a better measure for classifier

efficiency than the tradition classification accuracy, especially for imbalanced data sets.

$$\text{Areaunderthecurve(AUC)} = \frac{1}{2}\left[\frac{tp}{tp+fn} + \frac{tn}{tn+fp}\right] \quad \text{5-7}$$

- **Area under the curve combined with accuracy**

  This is a new, finer measure that makes use of some other methods in evaluating performance. It is correlated with the root mean square error.

$$\text{AUC(ACC)} = \frac{sensitivity+specificity}{2*Accuracy} \quad \text{5-8}$$

Table 5.3 and Table 5.4 present the classification results for the five data sets using two fitness measures ACC and AUC. The accuracy in both cases is comparable except for GER and HEP data. The reason is imbalanced class distribution in these datasets. Here, accuracy does not represent better discrimination power of a classifier resulting in lesser value of AUC. In Table 5.4 we can see that better value of AUC has also resulted in better accuracy, but vice versa is not true for imbalanced datasets. Therefore, the cases where the data is nearly balanced one cannot observe much difference in ACC and AUC of classifiers. As mentioned earlier, the reported results are average of best individuals obtained in 100 executions for each dataset after 250 generations.

Table 5.3 Results achieved using accuracy as fitness function

Data sets	ACC	Sensitivity	Specificity	AUC	AUC(ACC)
AUS	90.8	0.90	0.91	0.90	1.00
GER	79.0	0.25	1.00	0.62	0.79
HRT	82.7	0.76	0.84	0.80	0.99
CRD	86.3	0.85	0.86	0.86	1.00
HEP	81.0	0.91	0.71	0.80	0.99

Table 5.4 Results achieved using AUC as fitness function

Data sets	ACC	Sensitivity	Specificity	AUC	AUC(ACC)
AUS	90.00	0.86	0.93	0.90	1.00
GER	90.00	0.71	0.94	0.83	0.92
HRT	81.25	0.80	0.80	0.80	0.99
CRD	87.32	0.86	0.89	0.87	1.00
HEP	82.20	0.83	0.78	0.81	0.98

Table 5.5 presents an example two layered rule for German credit data set. In this rule there are two leaf nodes in outer layer, one of them is an arithmetic expression and other one is logical expression.

Table 5.5 Two layered rule for GER Data set

Outer Layer	(NOT (T1))OR (T2) (outer layer)
T1	A15="A151" Logical Inner Tree
T2	(((A18 / A8) / A2) + (A5 * (A18 - A16)) * A13) Arithmetic Inner Tree

The results of proposed methodology are compared with other GP based classification approaches. One can see that our approach yields compatible results with respect to accuracy. This type of comparison is usually not encouraged but as mentioned in previous chapters. GP is computationally expensive and requires very long training times. Therefore, experimental comparison with all the techniques is not possible. To overcome the problem of different partitions of data being used for experimentation, we have increased the rigorousness of our approach to make the results compatible. In the presented comparison the techniques like [58, 70] involve preprocessing and the methods presented in [32,104] use different representations. It is quite evident that our approach yields compatible results with respect to accuracy. Moreover, excessive bloat has been avoided by using DepthLimited crossover in which the classifiers cannot exceed the maximum depth of 8 which has been defined initially.

Table 5.6 Comparison of classification results with other techniques

Data set	L2GP (Average Accuracy)	Others
AUS	90.57 %	90.0% [104], 86.0% [70], 88.0% [105], 83.4% [102], 86.3% [30], 88.1% [58], 88.3% [101], 85.1% [32]
GER	77.9 %	78.0% [104] , 76.0% [105], 72.9% [30], 77.4% [101]
HR	82.8 %	86.0% [105], 78.7% [30], 77.9% [25], 82.2% [36]
CRD	87.2 %	84.8% [25] , 96.9% [106], 84.7% [36]

5.5 Conclusion

In this chapter a robust GP based classification technique is proposed that is applicable for data with mixed types of attributes. The technique does not require any transformation or preprocessing of the data and is readily applicable to all data types. The system is tested on several benchmark datasets and the performance is compared with various GP based classification methods. The results have revealed that the presented technique offers compatible performance with its more flexible two layered representation approach.

Chapter 6. Multi-Class Classification

6.1 Introduction

This chapter presents a new scheme for applying GP to multi-class classification problems. As mentioned in earlier chapters, GP possesses outstanding features like transparency, data modelling, flexibility and feature extraction, making it ideal for the classification task. However GP requires long training times along with some other issues. The proposed technique evolves significantly smaller number of classifiers as compared to binary decomposition method. This method also reduces the chances of conflict. The effectiveness of proposed technique is demonstrated over several benchmark datasets from UCI repository.

6.2 Related Work

Multi-class classification problems are common in real world applications for the tasks like object recognition, character recognition, person recognition, disease diagnosis and several others. Many of the classification algorithms are binary in nature and must be extended to be used for multi-class classification purpose. These include neural networks, decision trees, k-nearest neighbor, naive bayes classifiers, and support vector machines [107]. GP is one of these algorithms that are inherently binary in nature. Several methods have been presented to use GP for multi-class classification problems. Most noticeable among them are

1) Range selection method.
2) Binary decomposition method

Range selection method is applicable to GP classifiers that output real values. These classifiers are usually evolved for real and numerical data. This method can be extended to use nominal or categorical attributes by using some form of numeric encoding. The method works by declaring ‘c’ thresholds for ‘c’ class classification problems such as

$$\textbf{Class} = \begin{cases} \text{class 1}, & r < t_1 \\ \text{class 2}, & t_1 < r < t_2 \\ \vdots & \\ \text{class c}, & t_{c-1} < r < t_c \end{cases} \qquad 6\text{-}1$$

The range selection method requires optimal threshold selection which becomes challenging as the number of classes increases. The range selection mechanism includes assigning static thresholds [45, 46], dynamic thresholds [47, 46] and slotted thresholds [48].

In binary decomposition method which is also named as 'one versus all' method, one classifier is trained to recognize samples belonging to a particular class and reject all other samples. This results in 'c' classifiers for a 'c' class classification problem. The primary drawback of this method is that the training of many classifiers increases the computation. Moreover, the multiple classifiers may result in conflicts which increase proportionally with the number of classes. This results in increase of misclassification rate. Binary decomposition methods have been explored in [49-53].

Several other methods like 'all versus all' [108], error correcting output codes [109], and generalized error correcting output codes [110] have also been used to tackle multi-class classification problems by binary classification algorithms. However, none of them have been used in GP due to large number of computations involved.

Kishore et al [49] introduced the use of binary decomposition for multi-class classification using GP. The data can appear skewed when 'c' class classification problem is converted to binary class classification problem to evolve classifier for a particular class. This problem is solved by placing the instances belonging to the particular class alternately between samples belonging to other classes. This increases the number of training instances, increasing the time to evaluate the fitness of each individual in the population, which, in turn, adds to the overall complexity of the GP system. As per Kishore, if the number of instances belonging to a particular class= 'n' and the number of instances belonging to other classes='p', where p>>n, the data is increased to 'p-n' times such that number of instances of p is equal to number of instances of n, this increases each classifier evaluation 'p-n' times. The fitness function is the number of instances correctly classified or the classification accuracy of increased dataset.

$$\text{Fitness function} = \frac{\text{correct}}{[(p + n) + (p - n)]} \qquad 6\text{-}2$$

This method cannot measure the discriminating ability of classifier.

6.3 Proposed Binary Encoding

In this chapter a new binary encoding method for multi-class classification problems is introduced. In the proposed algorithm each of the 'c' classes is assigned a unique binary code with 'n' columns. Where

$$n = \text{ceil}\,(\log_2(C)) \qquad 6\text{-}3$$

Table 6.1 Classification matrix for a 4 class problem (binary encoding)

	Classifier 1	Classifier 2
Class1	0	0
Class2	0	1
Class3	1	0
Class4	1	1

This encoding results in a Classification Matrix (CM) that has '*N*C*' dimensions. For classification, the output codeword is compared to the codeword for the 'c' classes, and the matching codeword is considered the winning class. For instance, for a 4 class problem where N = 2, the coding is shown in Table 6.1.

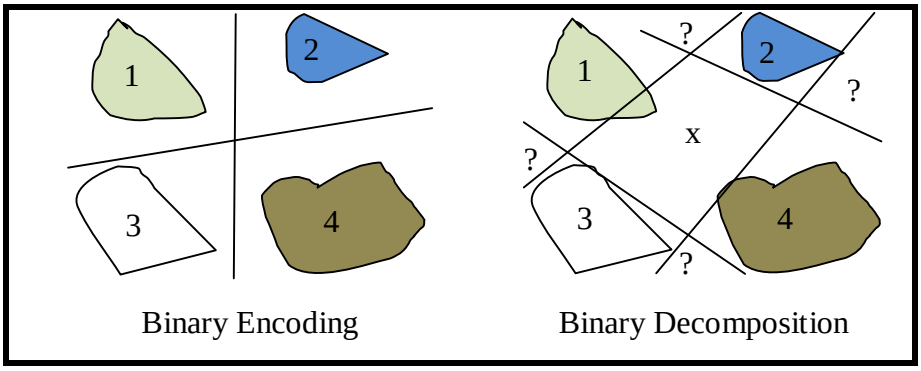

Figure 6-1Two approaches to multi-class classification

Figure 6-1 shows an example of an ideal 4 class classification problem, where two classifiers can effectively classify a 4 class classification problem using binary encoding method. On the other hand 4 classifiers will result in undecidable (x) and conflicting (?) regions, deteriorating the

overall classification performance. The new method offers numerous advantages over other GP based multi-class classification methods present in the literature.

The fitness function used in our approach can overcome the skewness of the data and favor the classifiers with better discriminative power. We do not increase the number of instances but get better discriminating estimate for a classifier for the same data. Using the variables introduced earlier, let 'cp' be the number of instances correctly classified by a classifier belonging to 'p'. and let 'cn' be correctly classified samples belonging to n.

The fitness function is :-

$$[(cn / n) + (cp / p)]*0.5, \qquad 6\text{-}4$$

where 0.5 is used to scale the value between 0 and 1.This ensures that the fitter classifier is a better discriminator between both pairs of classes. This is equivalent to area under the convex hull for one point '0'.

In case of binary decomposition method, the number of classifiers evolved, are equal to the number of classes in the data. There are two disadvantages associated with this scheme. First is the number of times the GP system is evolved, which increases with increase in number of classes linearly. This requires more computation resources using an algorithm (GP) that is already well known for its long training times. However, in the proposed encoding based classification method the number of classifiers evolved for a 'c' class classification problem are $n=ceil\ (\log_2(c))$. When compared to binary decomposition method, this number is smaller than the 'c' and decreases logarithmically with the increase in number of classes, requiring less GP evolutions. Binary Decomposition based classification results in conflicts between numerous classifiers. These conflicts increase with the increase in number of classes in the data, decreasing the classification accuracy. The proposed GP system can have at most 2^n-c conflicts. This is less than c conflicts faced by binary decomposition method. In the ideal case where $c=2^n$, there will be no conflict in proposed methodology. Our method is equally applicable to logical classifier expressions that output a boolean value and arithmetic classifier expressions that output a real value.

The algorithm for binary encoding based multi-class classification is mentioned in Algorithm 6-1.

```
Step 1. Begin
Step 2. C =  number of classes in the data
Step 3. N=ceil(log2(C))
Step 4. Initialize classification Matrix CM=(N*C) using binary encoding
Step 5. For 1 to n
            a. Initialize population
            b. Evaluate fitness using CM and N(Algorithm Fitness)
            c. Evolve population
            d. Output best
Step 6. End for
Step 7. Combine N classifiers
Step 8. Check training accuracy
Step 9. End
```

Algorithm 6-1 Encoding in GP (ENGP) for classification

6.3.1 Initialization

Just like other evolutionary algorithms the initial individuals in GP are randomly generated. Several algorithms exist to randomly initialize the trees in GP. The initialization method used by us is the well known *Ramped half and half* method [12]. The ramped half and half method utilizes advantages of both full and grow initialization schemes with equal probability. This method has been widely used for initialization in many classification problems [49, 54].The method has been found efficient in search for optimal solutions in GP.

6.3.2 Operators and Selection

The three operators used for evolution are reproduction, mutation and crossover. Reproduction selects a population member using fitness proportionate selection and copies it into the new population. Mutation operator chooses a parent randomly and selects a random node from the parent. This node is replaced by another random node of same type. i.e. a function node is replaced by a function node and a terminal node is replaced by a random terminal node. The crossover operator selects two parents using tournament selection and swaps randomly selected subtrees among them. DepthLimited crossover presented in chapter 3 is used to avoid increase in code size during evolution.

```
Step 1. Begin
Step 2. int countpositive,positive,countnegative,negative;
        a. For all instances in the training data inst
             i.  For all classes in the data c
                    1. For classifier n
                           Evaluate n for inst = e
                           If(CM[c,n]=1)
                                  Positive++
                                  If(e>=0)
                                       Countpositive++;
                           If(CM[c,n]=0)
                                  Negative++
                                  If(e<0)
                                       Countnegative++;
                    2. End classifier
            ii.  End classes
        b. End instances
Step 3. Fitness =[(countpositive/positive)+(countnegative/negative)]*0.5;
Step 4. Return Fitness
Step 5. End
```

Algorithm 6-2 Fitness for Encoding based GP

6.3.3 Fitness Function

The fitness algorithm is shown in Algorithm 6-2. As mentioned earlier the number of GP runs will be equal to the number of columns in the classification matrix. For each classifier each row corresponding to each class has an associated value. Every individual is data instance is evaluated against each class and values are checked in its corresponding column. If the value is positive and the corresponding cell is true, the true positive is incremented. True negative is incremented if the value is negative and cell value is false. This is the core idea in calculating the fitness function given in Algorithm 6-2.

6.4 Results

6.4.1 Experimental setup

The datasets used for experiments have been taken from UCI ML repository [84]. The properties of all data sets are given in Table 6.2. The data sets vary in number of classes, attributes and

instances. This has been done to evaluate the applicability and robustness of proposed classification scheme.

Table 6.2 Datasets used for experimentation

Datasets	Classes	Attributes	Type	Instances
IRIS	3	4	Real	150
WINE	3	13	Integer, Real	178
VEHICLE	4	18	Integer	946
GLASS	6	10	Real	214
YEAST	10	8	Real	1484

All the presented results have been averaged for tenfold cross validation applied 10 times on five different partitioning of the data.

The testing accuracies of all datasets are shown in Table 6.3. It can be noted that the proposed Binary Encoding based GP (ENGP) has outperformed binary decomposition based GP (BDGP) for all datasets.

Table 6.3 Accuracy for all datasets

Datasets	Algorithms	BDGP	ENGP
IRIS	ACC	96.00%	97.20%
	SD	0.2	0.3
	Classifiers	3	2
WINE	ACC	78.50%	90.10%
	SD	0.2	0.2
	Classifiers	3	2
VEHICLE	ACC	49.00%	83.00%
	SD	0.3	0.2
	Classifiers	4	2
GLASS	ACC	54.70%	76.70%
	SD	0.4	0.2
	Classifiers	6	3
YEAST	ACC	57.00%	73.80%
	SD	0.6	0.1
	Classifiers	10	4

The ACC column shows the accuracy, SD column shows the standard deviation and classifiers column shows the number of classifiers evolved for each datasets using BDGP and ENGP. One can observe a noticeable difference in performance of both algorithms and ENGP clearly outperformed BDGP.

6.4.2 Comparison with Other GP Approaches

Table 6.4 compares the performance of proposed ENGP with other GP based classification methods and it can be observed that the proposed algorithm has outperformed other methods in classification accuracy.

Table 6.4 Comparison of accuracies

Datasets	BDGP	ENGP	Others
IRIS	96%	97.2	96.6% [54], 97.3% [111]
WINE	78.5%	90.1	85.4%, 88.21%, 90.8%, 89.9% [46]
VEHICLE	49.08%	83.1	48.6% [54], 62.2% [50], 58.6% [111]
GLASS	54.7%	76.7	58% [32]
YEAST	57.05%	73.9	58.1% [112]

6.4.3 Convergence Behavior

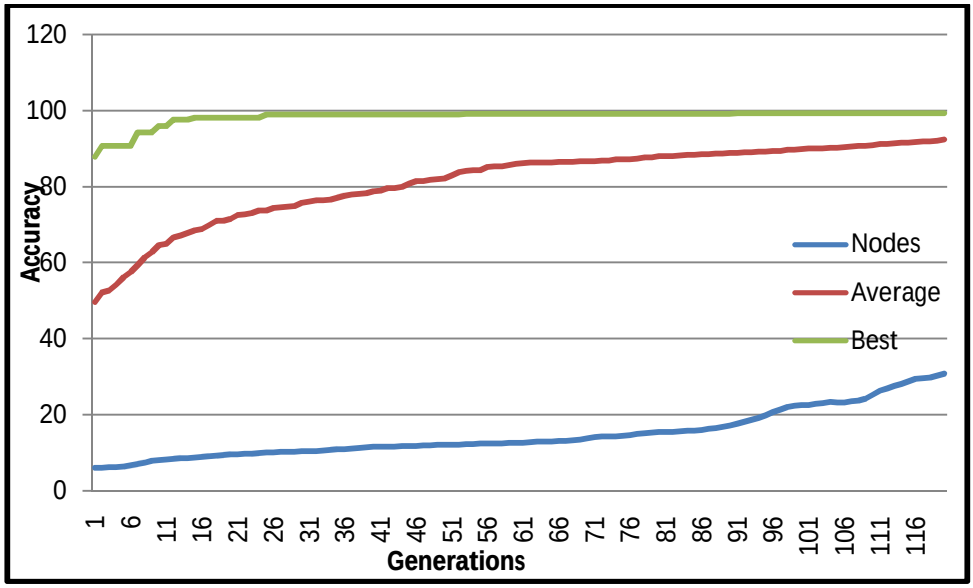

Figure 6-2 Number of nodes and fitness (Average, Best) for IRIS

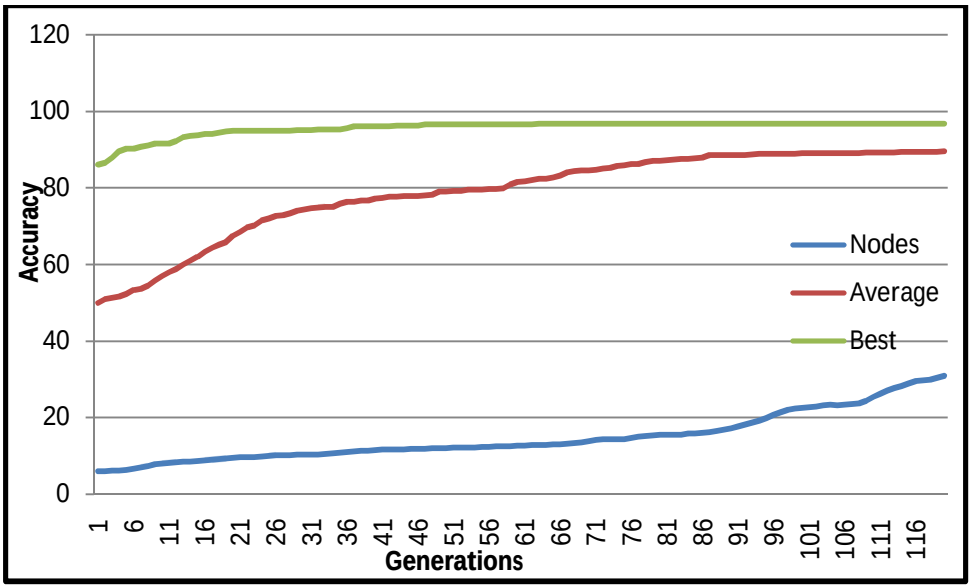

Figure 6-3 Number of nodes and fitness (Average, Best) for WINE

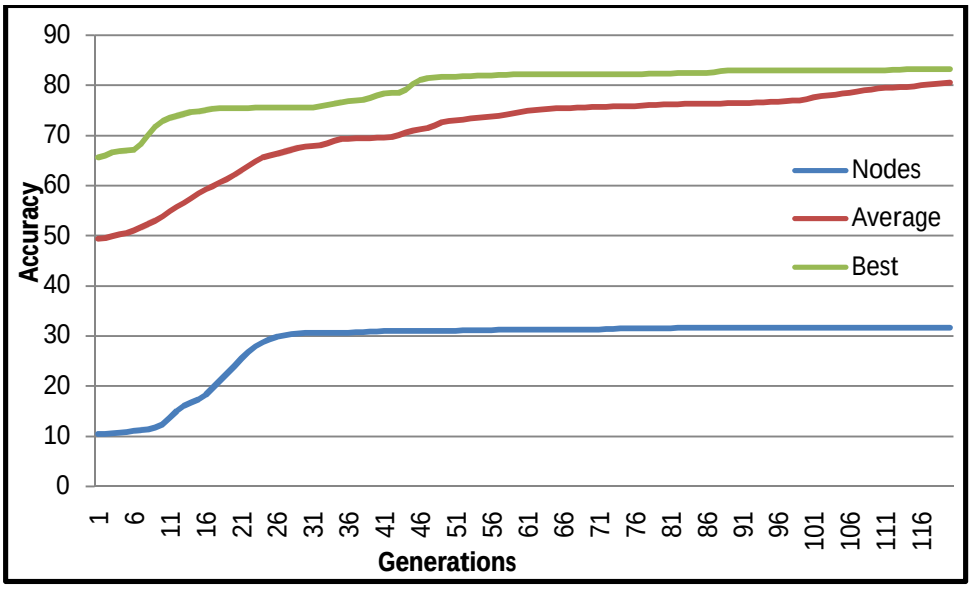

Figure 6-4 Number of nodes and fitness (Average, Best) for VEHICLE

Figure 6-2 presents the convergence behavior of proposed scheme for IRIS. It is visible that the size of classifiers does not increase beyond defined maximum depth. The average fitness and best fitness during the evolution process is reported. The results presented in the graph below are averaged for 10 executions of tenfold cross validation of ENGP. Figure 6-3 presents the convergence behavior of GP system to WINE dataset with same experimental settings.

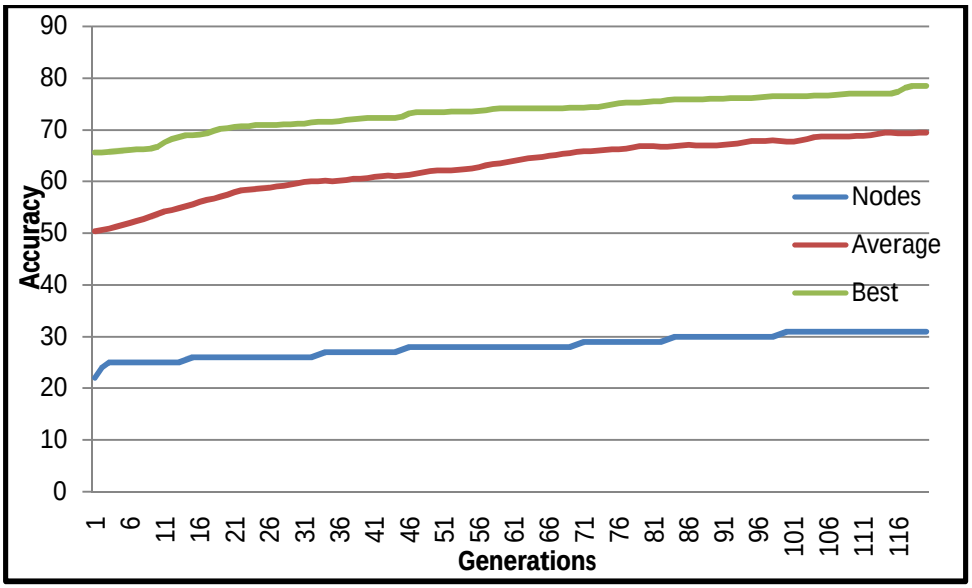

Figure 6-5 Number of nodes and fitness (Average, Best) for GLASS

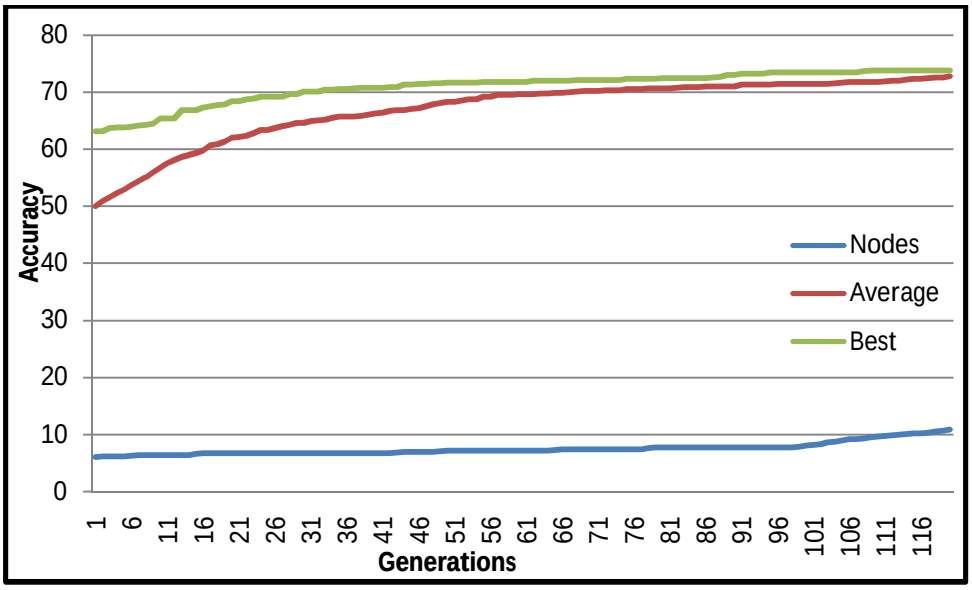

Figure 6-6 Number of nodes and fitness (Average, Best) for YEAST

Figure 6-5 and Figure 6-7 present convergence behavior of proposed GP system for GLASS and YEAST dataset. We can see a controlled increase in average number of nodes due to use of depth Limited crossover, and a steady convergence of the population.

6.5 Conclusion

Our proposed methodology has effectively addressed the issue of long training time implicitly by reducing the number of evolutions involved for multi-class classification problems. The proposed binary encoding method has been tested against several benchmark datasets and good results have been found. It not only outperformed the traditional binary decomposition method in terms of classification accuracy but also proved to be a more efficient method by using less number of classifiers. This methodology has a huge potential for further research and it can be extended to be used for other binary classifiers.

Chapter 7. Conclusions and Future Work

The aim of this thesis, was to investigate and optimize the use of GP for classification tasks. GP, despite being used for classification since long due to its numerous outstanding abilities, still suffers from a few limitations. In this thesis we have addressed these issues and proposed some optimizations for GP based classification.

The problems that have been addressed are:

1. Increase in complexity during evolution that adds to long training time.
2. Lack of convergence to same solution. In GP, the solutions can differ in size and fitness for each evolution.
3. Non applicability of GP based classification methods for mixed data-type. The data must be transformed into one or other type for mixed variable classification.
4. Lacking an efficient method for multi-class classification.

In this thesis we have optimized the GP based classification approach and proposed methodologies to overcome the above mentioned drawbacks. The proposed solutions are as follows.

7.1 Taxonomy of Classification using GP

Taxonomy of classification approaches using GP has been presented that groups the classification techniques based upon the strategy used to evolve and represent the classifiers. Lots of work has been done for classification using GP but this is the first attempt to provide a taxonomy and comprehensive over view of all the approaches.

7.2 DepthLimited Crossover

DepthLimited Crossover operator has been proposed for data classification problems to avoid inefficient increase in code size of classifiers,. This technique does not let trees increase in complexity while still maintaining diversity and efficiency of search during evolution. The

performance of traditional GP and DepthLimited crossover has been compared for data classification problem and found that DepthLimited crossover technique provides compatible results without expanding the search space beyond initial limits. Initial depth limits have been defined as $depth_p = ceil(\log_2(A_d))$ where A_d is the number of attributes in the dataset. This limit needs to be customized and carefully defined for different problems in order to ensure highly efficient search mechanism.

7.3 PSO based Optimization

To optimize GP evolved classifiers, a hybrid approach to data classification has been presented which eliminates the need for evolving classifiers for longer number of generations and optimizes the evolved classifiers to achieve better performance. This hybrid classification technique is a two step process. ACE is evolved using GP in the first step and weights are appended to this expression along all the terminals. These weights form a swarm particle. In the second step, a population of random weight particles is optimized using Particle Swarm Optimization (PSO). This combines GP and PSO hybrid classification technique (GPSO). The empirical comparison of GP and GPSO has been performed on benchmark data classification problems. GPSO has successfully yielded promising results and outperformed GP in terms of accuracy and Number of Function Evaluations (NFE) for all datasets in training as well as testing process. The future research directions include use of proposed tuning algorithm in other variants of GP used for classification. Some other optimization algorithms can also be explored for optimization of added weights.

7.4 Mixed Type Data Classification

For classification of mixed type data, a two layered approach has been proposed where the outer layer contains logical functions and some nodes which form the inner layer and are either logical or arithmetic expressions. Logical expressions output a boolean value used by the outer tree. The arithmetic expressions output a real value. Positive real value is considered true and negative value is considered false for the outer logical layer tree. The technique has been applied on various classification problems and found successful. This technique can be extended for multi-class classification problems.

7.5 Binary Encoding Based Method

To tackle the problem of multi-class classification problem, a binary encoding based method has been presented. The proposed technique evolves $c = ceil(\log_2(n))$ classifiers for 'n' class classification problem. This number is petite as compared to 'one versus all' (binary decomposition) method usually used for GP based multi-class classification. This method also reduces the chances of conflict. The conflicts in binary decomposition method can be at most 'n' where as in case of proposed method it cannot exceed 2^c-n. There can be no conflict if $n=2^c$. The effectiveness of proposed technique over several benchmark datasets from UCI repository has been presented.

7.6 Addition to Existing Literature

GP lacks considerable attention to optimize this scheme for classification tasks. That was the primary reason why I targeted the optimization of GP for classification. The aim was to increase its feasibility for a task it is already good at, by removing a few limitations.

The work done in this thesis adds a solution to all of the most commonly known problems with GP based classification approach. Through this work, we have successfully demonstrated the significant reduction in the bloat problem for classification task by presenting a hybrid approach to increase performance of GP evolved classifiers. The approach offers better performance in fewer computations. This work also introduces a new two layered approach for mixed type data classification with an added feature that uses data in its original form instead of any transformation or preprocessing. Another value addition of this thesis is an encoding method, for multi-class classification problem, that evolves fewer classifiers, reducing the computation involved and suffers from fewer conflicts giving better results.

All these methods have added efficiency to GP for classification purpose making it a better choice for classification problems.

7.7 Future Research

This work has opened grounds for future research in the field of GP optimization. Most of the presented methodologies can be explored further for use in many other disciplines.

The proposed taxonomy can be extended to include linear GP, parallel GP and cellular GP based approaches to provide an exhaustive overview.

The presented 'DepthLimited Crossover' has the potential to be explored for numerous other problems where GP offers an efficient solution. Some dynamically increasing factors, that can allow population to increase its size beyond the initial limits, can be used, if a desired solution is not found within a limited number of generations.

The proposed 'PSO based optimization' can also be complemented and tested with several other optimization methods present in the literature in order to be explored for further. Various other schemes for optimization like adding weights to all edges and optimizing them using back propagation, can also provide some future research topics for researchers.

Investigating two layered approach for multi-class classification is part of an ongoing work. It can be researched into further to investigate the addition of more functions providing more flexibility to the classifiers. Another vital research arena is the elimination of undecidable regions for multi-class classification.

Overall, the work proposed in this thesis has added the possibility for the optimization of data classification process using GP, highlighting its benefits and getting rid of its demerits, increasing the overall effectiveness and efficiency.

References

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

[2] I W Flockhart and N J Radcliffe, "GA-MINER: Parallel Data Mining with Hierarchical Genetic Algorithms," University of Edinburgh, 1995.

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

[4] B Garcia, R Aler, A Ledezma, and A Sanchis, "Protein-Protein Functional Association Prediction Using Genetic Programming," in *GECCO '08: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*, 2008.

[5] P Widera, J M Garibaldi, and N Krasnogor, "Evolutionary Design of the Energy Function for Protein Structure Prediction," in *2009 IEEE Congress on Evolutionary Computation*, 2009.

[6] Y Inagaki, "On Synchronized Evolution of the Network of Automata," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 147-158, 2002.

[7] J J Kim and B T Zhang, "Effects of Selection Schemes in Genetic Programming for Time Series Prediction," in *Proceedings of the 1999 Congress on Evolutionary Computation, CEC 99*, 1999.

[8] R Fry and A Tyrrell, "Enhancing the Performance of GP using an Ancestry-Based Mate Selection," in *Proceedings of Genetic and Evolutionary Computation GECCO 2003*, 2003.

[9] V Podgorelec, P Kokol, B Stiglic, and I Rozman, "Decision Trees: An Overview and Their Use in Medicine," *Journal of Medical Systems*, vol. 26, no. 5, October 2002.

[10] N L Cramer, "A Representation for the Adaptive Generation of Simple Sequential Programs," in *Proceedings of International Conference on Genetic Algorithms and the Applications*, 1985.

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

[12] J R Koza, "Genetic Programming: On the Programming of by Means of Natural Selection," in *MA*, Cambridge, 1992.

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

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

[15] J R Koza, "Concept Formation and Decision Tree Induction using the Genetic Programming Paradigm," in *Parallel Problem Solving from Nature,Proceeding of first workshop,Lecture Notes in Computer Science*, 1991.

[16] Q Li, M Yao, W Wang, and X Cheng, "Dynamic Split-Point Selection Method for Decision Tree Evolved by Gene Expression Programming," in *IEEE Congress on Evolutionary Computation*, 2009.

[17] R E Marmelstein and G B Lamont, "Pattern Classification using a Hybrid Genetic Program – Decision Tree Approach," in *Proceedings of the Third Annual Conference of Genetic Programming*, 1998, pp. 223-231.

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

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

[20] M Bot, "Application of Genetic Programming to Induction of Linear Classification Trees," Faculty of Exact Sciences, Vrije Universiteit, Amsterdam, Final Term Project Report 1999.

[21] A P Engelbrecht, L Schoeman, and S Rouwhorst, "A Building Block Approach to Genetic Programming for Rule Discovery, in Data Mining: A Heuristic Approach," in *Data Mining.*: Idea Group Publishing, pp. 175-189.

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

[23] G Folino, C Pizzuti, and G Spezzano, "Parallel Genetic Programming for Decision Tree Induction," in *Proceedings of the 13th International Conference on Tools with Artificial Intelligence*, Dallas, TX USA, 2001.

[24] A Tsakonas, "A Comparison of Classification Accuracy of Four Genetic Programming-Evolved Intelligent Structures," *Information Sciences*, pp. 691-724, 2006.

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

[26] M Oltean and L Diosan, "An Autonomous GP-based System for Regression and Classification Problems," *Applied Soft Computing*, vol. 9, pp. 49-60, 2009.

[27] J R Quinlan,. San Francisco: Morgan Kaufmann.

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

[29] A A Freitas, "A Genetic Programming Framework for Two Data Mining Tasks : Classification and Generalized Rule Induction," in *Genetic Programming*, CA , USA, 1997, pp. 96-101.

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

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

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

[33] J J Huang, G H Tzeng, and C S Ong, "Two-Stage Genetic Programming (2SGP) for the Credit Scoring Model," *Applied Mathematics and Computation*, vol. 174, no. 2, pp. 1039-1053, 2006.

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

[35] F J Berlanga, M J Jesus, M J Gacto, and F Herrera, "A Genetic-Programming-Based Approach for the Learning of Compact Fuzzy Rule-Based Classification Systems," *Lecture Notes on Artificial Intelligence (LNAI)*, pp. 182-191, 2006.

[36] R R F Mendes, F B Voznika, A A Freitas, and J C Nievola, "Discovering Fuzzy

Classification Rules with Genetic Programming and Co-Evolution," *Lecture notes in Artificial Intelligence*, pp. 314-325, 2001.

[37] B C Chien, J Y Lin, and T P Hong, "Learning Discriminant Functions with Fuzzy Attributes for Classification using Genetic Programming," *Expert Systems with Applications*, vol. 23, no. 1, pp. 31-37, 2002.

[38] C C Bojarczuk, H S Lopes, A A Freitas, and E L Michalkiewicz, "A Constrained-Syntax Genetic Programming System for Discovering Classification Rules: Application to Medical Datasets," *Artificial Intelligence in Medicine*, pp. 27-48, 2004.

[39] C C Bojarczuk, H S Lopes, and A A Freitas, "An Innovative Application of a Constrained-Syntax Genetic Programming System to the Problem of Predicting Survival of Patients," *Lecture Notes in Computer Science*, vol. 2610, 2003.

[40] A Tsakonas, G Dounias, J Jantzen, H Axer, and B Bjerregaard, "Evolving Rule-based Systems in Two Medical Domains using Genetic Programming," *Artificial Intelligence in Medicine*, pp. 195-216, 2004.

[41] J Y Lin, H R Ke, B C Chien, and W P Yang, "Classifier Design with Feature Selection and Feature Extraction using Layered Genetic Programming," *Expert Systems with Applications*, vol. 34, pp. 1384-1393, 2007.

[42] E Carreno, G Leguizamon, and N Wagner, "Evolution of Classification Rules for Comprehensible Knowledge Discovery," in *IEEE Congress on Evolutionary Computation*, 2007, pp. 1261-1268.

[43] J Eggermont, A E Eiben, and J I Hemert, "A Comparison of Genetic Programming Variants for Data Classification," in *Proceedings of the Eleventh Belgium Netherlands conference on Artificial Intelligence*, 1999, pp. 253-254.

[44] R Konig, U Johansson, and L Niklasson, "Genetic Programming - A Tool for Flexible Rule Extraction," in *IEEE Congress on Evolutionary Computation*, 2007.

[45] M Zhang and V Ciesielski, "Genetic Programming For Multiple Class object Detection," in *Proceedings of the 12th Australian Joint Conference on Artificial Intelligence*, Australia, 1999, pp. 180–192.

[46] D Parrott, X Li, and V Ciesielski, "Multi-objective Techniques in Genetic Programming for Evolving Classifiers," in *IEEE Congress on Evolutionary Computation*, 2005, pp. 183–190.

[47] W R Smart and M Zhang, "Classification Strategies for Image Classification in Genetic Programming," in *Proceeding of Image and Vision Computing NZ International Conference*, 2003, pp. 402-407.

[48] M Zhang and W Smart, "Multiclass Object Classification using Genetic Programming," *Lecture Notes in Computer Science*, pp. 367–376, 2004.

[49] J K Kishore, L M Patnaik, A Mani, and V K Agrawal, "Application of Genetic Programming for Multicategory Pattern Classification," *IEEE transactions on Eolutionary Computation*, 2000.

[50] T Loveard and V Ciesielski, "Representing Classification Problems in Genetic Programming," in *IEEE Congress on Evolutionary Computation*, 2001, pp. 1070-1077.

[51] W Smart and M Zhang, "Using Genetic Programming For Multiclass Classification By Simultaneously Solving Component Binary Classification Problems," *Lecture Notes in Computer Science*, 2005.

[52] A Teredesai and V Govindaraju, "Issues in Evolving GP based Classifiers for a Pattern Recognition Task," in *IEEE Congress on Evolutionary Computation*, 2004, pp. 509-515.

[53] C C Bojarczuk, H S Lopes, and A A Freitas, "Genetic Programming for Knowledge Discovery in Chest-Pain Diagnosis," *IEEE Engineering in Medicine and Biology Magazine*, pp. 38-44, 2000.

[54] D P Muni, N R Pal, and J Das, "A Novel Approach To Design Classifiers Using GP," *IEEE Transactions of Evolutionary Computation*, 2004.

[55] R R F Mendes, F B Voznika, A A Freitas, and J C Nievola, "Discovering Fuzzy Classification Rules with Genetic Programming and Co-Evolution," in *Genetic and Evolutionary Computation Conference, Late Breaking Papers*, 2001.

[56] M Ostaszewski, P Bouvry, and F Seredynski, "Multiobjective Classification with moGEP: An Application in the Network Traffic Domain", in *Genetic and Evolutionary Computation Conference*, pp. 635-642, ACM, 8-12 July 2009.

[57] P Lichodzijewski and M I Heywood, "Pareto-Co evolutionary Genetic Programming for Problem Decomposition in Multi-Class Classification," in *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, London, 2007.

[58] Khalid Badran and Peter Rockett, "Integrating Categorical Variables with Multiobjective Genetic Programming for Classifier Construction," in *European Conference on Genetic*

Programming, LNCS, 2008.

[59] S Luke, "Code Growth is Not Caused by Introns," in *Genetic and Evolutionary Computation Conference*, 2000, pp. 228-235, Late Breaking Papers.

[60] P G Espejo, S Ventura, and F Herrera, "A survey on the application of genetic programming to classification," *EEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 40, no. 2, pp. 121-144, 2010.

[61] C Zhou, W Xiao, T M Tirpak, and P C Nelson, "Evolving Accurate and Compact Classification Rules With Gene Expression Programming," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 6, pp. 519-531, 2003.

[62] T Loveard, "Genetic Programming Methods for Classification Problems," Department of Computer Science, RMI, PhD Thesis 2003.

[63] S M Winkler, M Affenzeller, and S Wagner, "Using Enhanced Genetic Programming Techniques for Evolving Classifiers in the Context of Medical Diagnosis," *Genetic Programming and Evolvable Machines*, pp. 111-140, 2009.

[64] T Ito, H Iba, and S Sato, "Depth-Dependent Crossover for Genetic Programming," in *Proceedings of the 1998 IEEE World Congress on Computational Intelligence*, Alaska, 1998, pp. 775-780.

[65] H Jabeen and A R Baig, "DepthLimited Crossover in Genetic Programming for Classifier Evolution," *Computers in Human Behaviour*, 2010, To Appear.

[66] M Zhang and P Wong, "Genetic Programming for Medical Classification: A Program Simplification Approach," *Genetic Programming and Evolvable Machines*, pp. 229-255, 2008.

[67] A Topchy and W F Punch, "Faster Genetic Programming based on Local Gradient Search of Numeric Leaf Values," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2001, pp. 155-162.

[68] M Zhang and W Smart, "Genetic Programming with Gradient Descent Search for Multiclass Object Classification," in *7th European Conference on Genetic Programming, EuroGP*, 2004, pp. 399-408.

[69] M Zhang and W R Smart, "Learning Weights in Genetic Programs Using Gradient Descent for Object Recognition," in *Applications of Evolutionary Computing, EvoWorkshops*, 2005, pp. 417-427.

[70] T Loveard and V Ciesielski, "Employing Nominal Attributes in Classification using Genetic Programming," in *4th Aisa pacific conference on simulated evolution and learning*, singapore, 2002, pp. 487-491.

[71] W B Langdon and R Poli, "Fitness Causes Bloat," School of computer science, University of Birmingham, CSRP-97-09, 1997.

[72] T Soule, "Code Growth in Genetic Programming," college of graduate studies, University of Idaho, 1998.

[73] M Brameier and W Banzhaf, "Neutral Variations Cause Bloat in Linear GP," in *European conference on Genetic Programming*, 2003, pp. 286-296.

[74] R Poli, "A Simple but Theoretically-Motivated Method to Control Bloat in Genetic Programming.," in *European Conference on Genetic Programming*, , 2003.

[75] W B Langdon and W Banzhaf, "Parallel Problem Solving from Nature," in *6th Internation conference on Parallel Problem Solving from Nature,LNCS*, 2000.

[76] N F Mcfee and J D Miller, "Accurate Replication in Genetic Programming," in *International Conference on Genetic Algorithms*, 1995.

[77] S Gelly, O Teytaud, N Bredeche, and M Schoenauer, "A Statistical Learning Theory Approach of Bloat," in *Genetic and Evolutionary Computation Conference*, , 2005.

[78] T Soule and J A Foster, "Removal Bias: a New Cause of Code Growth in Tree Based Evolutionary Programming," in *IEEE International Conference on Evolutionary Computation*, 1998.

[79] S Dignum and R Poli, "Generalisation of the Limiting Distribution of Program Sizes in Tree-Based Genetic Programming and Analysis of its Effects on Bloat," in *Genetic and evolutionary computation Conference*, 2007.

[80] W A Tackett, "Recombination, Selection And The Genetic Construction Of Computer Programs," Department of electrical engineering systems, University of southern California, PhD Thesis 1994.

[81] P Dhaeseleer, "Context Preserving Crossover in Genetic Programming," in *Proceedings of the 1994 IEEE World Congress on Computational Intelligence*, Orlando, Florida, USA, 1994.

[82] H Majeed and C Rayan, "On the Constructiveness of Context-Aware Crossover," in

Conference on Genetic and evolutionary computation, 2007.

[83] H Majeed, "The Importance of Semantic Context in Tree Based GP and its Application in Defining a Less Destructive, Context Aware Crossover for GP," University of Limerick, PhD Thesis 2007.

[84] A Asuncion and D J Newman. (2007) Machine Learning Repository. [Online]. http://archive.ics.uci.edu/ml/

[85] B D Ripley. [Online]. http://www.stats.ox.ac.uk/pub/PRNN/

[86] J Kennedy and R C Eberhart, "Particle Swarm Optimization," in *IEEE International Conference on Neural Networks*, 1995, pp. 1942-1948.

[87] J S Seo et al., "Multimodal Function Optimization based on Particle Swarm Optimization," *IEEE Transactions on Magnetics*, 2006.

[88] V G Gudise and G K Venayagamoorthy, "Comparison of Particle Swarm Optimization and Back propagation as Training Algorithms for Neural Networks," in *Swarm Intelligence Symposium*, 2003.

[89] M Clerc and J Kennedy, "The Particle Swarm Explosion, Stability, and Convergence In A Multidimensional Complex Space," *IEEE Transaction on Evolutionary Computation*, pp. 58–73, 2002.

[90] Y Shi and R C Eberhart, "A Modified Particle Swarm Optimizer," in *IEEE Congress on Evolutionary Computation*, 1998, pp. 69–73.

[91] Y Shi and R C Eberhart, "Particle Swarm Optimization with Fuzzy Adaptive Inertia Weight," in *Proceedings of the Workshop on Particle Swarm Optimization*, Indianapolis, 2001.

[92] J Kennedy, "Small worlds and mega-minds: Effects of Neighborhood Topology on Particle Swarm Performance," in *In Proceedings of Congress on Evolutionary Computation*, 1931–1938, p. 1999.

[93] P N Suganthan, "Particle Swarm Optimizer with Neighborhood Operator," in *In Proceedings of Congress Evolutionary Computation*, Washington, 1999, pp. 1958-1962.

[94] K E Parsopoulos and M N Vrahatis, "UPSO: A Unified Particle Swarm Optimization Scheme," in *Lecture Series on Computational Sciences*, 2004, pp. 868–873.

[95] R Mendes, J Kennedy, and J Neves, "The Fully Informed Particle Swarm: Simpler, Maybe Better," *IEEE Transactions on Evolutionary Computation*, pp. 204–210, 2004.

[96] J Riget and J Vesterstrom, "A Diversity Guided Particle Swarm Optimizer - The ARPSO," Department of Computer Science, University of Aarhus, 2002.

[97] A Silva, A Neves, and E Costa, "Chasing The Swarm: A Predator Pray Approach to Function Optimization," in *International Conference on Soft Computing*, 2002.

[98] D Parrot and X Li, "Locating and Tracking Multiple Dynamic Optima by a Particle Swarm Model Using Speciation," *IEEE Transactions on Evolutionary Computation*, 2006.

[99] T M Blackwell and P J Bentley, "Dynamic Search with Charged Swarms," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2002.

[100] R Poli, J Kennedy, and T Blackwell, "Particle Swarm Optimization : An Overview," *Swarm Intelligence*, pp. 33-57, 2007.

[101] C S Ong, J J Huang, and G H Tzeng, "Building Credit Scoring Models using Genetic Programming," *Expert Systems with Applications*, vol. 29, no. 1, pp. 41-47, 2005.

[102] S Sakprasat and M C Sinclair, "Classification Rule Mining for Automatic Credit Approval Using Genetic Programming," in *IEEE Congress on Evolutionary Computation*, Singapore, 2007, pp. 548-555.

[103] M Sokolova and G Lapalme, "Performance Measures in Classification of Human Communications," *Lecture notes in artificial intelligence*, vol. 4509, pp. 150-170, 2007.

[104] J J Huang, C S Ong, and G H Tzeng, "Two-stage Genetic Programming (2SGP) for the credit scoring model," *Applied Mathematics and Computation*, vol. 174, no. 2, pp. 1039-1053, 2006.

[105] Y Zhang and P Rockett, "A Comparison of Three Evolutionary Strategies for MultiObjective Genetic Programming," *Artificial Intelligence Review*, vol. 27, pp. 149-163, 2007.

[106] A Baykasoglu and L Ozbakir, "MEPAR-miner: Multi-Expression Programming for Classification Rule Mining," *European Journal of Operational Research*, vol. 183, no. 2, pp. 767-784, 2007.

[107] M Aly, "Survey on Multiclass Classification Methods," 2005.

[108] T Hastie and R Tibshirani, "Classification by Pairwise Coupling," *Advances in Neural Information Processing Systems*, vol. 10, 1998.

[109] T G Dietterich and G Bakiri, "Solving Multiclass Learning Problems via Error Correcting Output Codes," *Journal of Artificial Intelligence Research*, pp. 1–38, 1995.

[110] E Allwein, R Shapire, and Y Singer, "Reducing multiclass to binary: A unifying approach for margin classifiers," *Journal of Machine Learning Research*, pp. 113-141, 2000.

[111] N Holden and A A Frietas, "A hybrid PSO/ACO Algorithm for Classification," in *Genetic And Evolutionary Computation Conference*, London, UK, 2007.

[112] V Athitsos and S Sclaroff, "Boosting Nearest Neighbor Classifiers for Multiclass Recognition," Computer Science, Boston University, 2004.

[113] J Han and M Kamber, *Data Mining: Concepts and Techniques*.: Morgan Kaufmann Publishers, 2006.

[114] J Eggermont, "Evolving Fuzzy Decision Trees for Data Classification," in *Proceedings of the 14th Belgium Netherlands Artificial Intelligence Conference*, 2002.

[115] J Demsar, "Statistical Comparisons of Classifiers over Multiple Data Sets," *Journal of Machine Learning Research*, pp. 1-30, 2006.

[116] C C Bojarczuk, H S Lopes, and A A Freitas, "Data mining with Constrained-Syntax Genetic Programming: Applications to Medical Datasets," in *Proceedings Intelligent Data Analysis in Medicine and Pharmacology*, 2001.

[117] H Jabeen and A R Baig, "A Framework for Optimization Of Genetic Programming Evolved Classifier Expressions," *Lecture Notes in Computer Science*, 2010. (to appear)

[118] H Jabeen and A R Baig, "Two Layered Genetic Programming for Mixed Variable Data Classification," *Applied Soft Computing*. (under review)

[119] H Jabeen and A R Baig, "Optimization of Genetic Programming Evolved Arithmetic Classifier Expressions using Particle Swarm Optimization for MultiClass Classification," *Knowledge Based Systems*. (under review)

[120] H Jabeen and A R Baig, "GPSO:Optimization of Genetic Programming Classifier Expressions for Binary CLassification using Particle Swarm Optimization,"*International Journal of Innovative Computing, Information and Control*. (under review)

[121] H Jabeen and A R Baig, "A Review of CLassification Using Genetic Programming," *International Journal of Engineering Science and Technology*, vol. 3, no. 2, 2010.

[122] H Jabeen and A R Baig, "Particle Swarm Optimization Based Tuning of Genetic Programming Evolved Classifier Expressions," in *Studies in Computational Intelligence*. Granada, Spain: Springer, 2010. (to appear)