Journal of Theoretical and Applied Information Technology
20th May 2013. Vol. 51 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

MULTI-OBJECTIVE GENETIC PROGRAMMING TECHNIQES FOR MULTI-CLASSIFICTION

[1]KHALED M. KRIDAN, [2]ALAA ROHIM, [3]KHALED BADRAN
E-mail: [1]kkridan@yahoo.co.in, [2]alaa_rohim@yahoo.com [3]khaledbadran@yahoo.co.in

ABSTRACT

In this paper, a generic, multi- classification with multi-objective genetic programming method is proposed to design an optimal feature extraction phase followed by a simple threshold classifier. This is implemented by projecting the pattern space into a decision space in which the discriminability between classes is maximized. Three multi-classification decomposition techniques (One-from-*n*, Hierarchical, Single feature space mapping SFSM) are applied to real world five datasets from the UCI Machine Learning database have been used to verify our approach. These methods are primarily focused on multi-classes feature extraction.

Keywords: *Genetic Programming(GP), Feature Extraction, Multi Classification*

1. INTRODUCTION

1.1 Multi-category Classification

Typically in pattern classification, feature extraction and selection is globally done, but in problems with many classes and many features, not all features will necessarily be good discriminators for all classes. Moreover, many classification algorithms are specifically designed to solve 2-class problems. For a *k*-class problem ($k > 2$) it can be divided into a set of 2-class problems that need to be recombined to make the overall decision for each pattern. With dividing the problem into a set of sub-problems, this process can be done either in *global* mode, where these set of feature is shared inside each sub-problem, or in *local* mode, where a separate feature set is selected for each sub-problem independently of all other sub-problem. Local mode may increase the classification accuracy but may also increase the computational cost for evaluating the features over all sub–problems – which is normally the principal computational cost of evolutionary training algorithms. Generally, three principal approaches (One-from-n, Hierarchical and SFSM) have been used for this decomposition which has been summarised in section 3:

1.2 Feature Extraction with Genetic program Multi-Category Classification

A major challenge in pattern recognition is the extraction (or construction) of new features from the raw data to produce a better classifier figure1. This can also reduce the computational complexity of classification process since many datasets contain large numbers of attributes, some of which have little effect on the classifier decision. Genetic programming has been applied to feature extraction and selection by transforming the original pattern vector into decision space. This mapping is implemented using the set of functions available to the evolution process, maybe linear, non-linear, logical or any special function set suitable for the application. It is not necessary (or indeed usual) for all of the original features to be selected as terminals for each GP tree which means the feature selection task is accomplished alongside the feature extraction task.

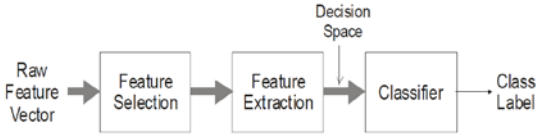

Figure1: Prototypical Pattern Recognition System

Each GP tree will map the input feature space to a single dimension represented by real values. Then, these output numeric values are translated into class labels. For single-dimension mapping, this translation can be either a simple threshold for binary classification, or use multiple pre-defined thresholds to form different regions in the transformed decision space indicating different classes. This way of labelling means GP is doing both the tasks of feature extraction and classification. The second approach is to use an independent classifier to dynamically divide the

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

output space into class regions. Many publications have been presented which employ GP to find feature extraction and selection stages combined with different types of classifiers.

1.3 Multi-Objective Genetic Programming

The chromosomes in a genetic programming (GP) population will 'bloat' – that is, grow without limit without any accompanying improvement in fitness. Research on the causes of bloat has been recently summarised in [1]. Indeed, Langdon and Poli[2]-[3] have shown that any variable-length representation suffers from bloating. The objections to bloat are similarly well-rehearsed: Excessive computation times and complex solutions which are held to generalise less well than more compact trees.

Multi-objective genetic programming techniques can be employed to overcome 'bloat' and to evolve programs for less computational cost and to generalise better than would be the case using the standard GP. Recently a lot of work has been published on using multi objective genetic programming in genuinely multi-objective optimisation applications done and have been recently surveyed in [1] but much of the use of multi objective optimisation techniques in GP has been aimed at controlling bloat.

Zhang and Rockett [4]-[5] applied multi-objective genetic programming (MOGP) to search the feature space under the selective pressure defined by multiple objectives driven by the Pareto optimality concept. First, they mapped the input feature space into a one-dimensional decision space. Then they used Golden Section search as a fast means of finding the optimal threshold of a classifier for binary classification problems.

The parsimony pressure approach is one approach used to solve bloat problem. It use a multi-objective method in which the (strictly) non-commensurable objectives of problem-specific error and tree complexity are handled in a Pareto optimisation framework [4]. Here, given two solutions of, say, equal error, the solution with the smaller number of nodes is said to *dominate* the larger solution, is more highly ranked and is thus assigned a better fitness value. It is hence more likely to be selected for subsequent breeding. What results from the Pareto framework is not a single unique solution but a set of equivalent solutions which lie on a Pareto front (or surface) in objective space and which delineate the fundamental trade-offs in the problem; no point on the Pareto front can be modified to improve one objective without simultaneously degrading another. Multi-objective GP (MOGP) has a number of advantages: As well as controlling bloat very effectively, it does not require a pre-determined depth-limit parameter and the tree depth is free to adjust to suit the problem at hand. Zhang and Rockett [4]-[5] have successfully used MOGP for a range of optimisations connected with pattern recognition problems. Rodríguez-Vázquez *et al.* [6] have also reported using MOGP on a systems identification problem in control engineering.

Recently a lot of work has been published on using multi objective genetic programming in genuinely multi-objective optimisation applications done and have been recently surveyed in [1] but much of the use of multi objective optimisation techniques in GP has been aimed at controlling bloat.

Rodriguez-Vazquez *et al.* [7] published a set of papers starting in 1997 in which they applied MOGP to non-linear system identification. Individuals' selection was based on the Pareto dominance concept with two objectives: Fitness and model complexity. Individuals were ranked, based on how many other individuals dominated them – fitness was based on their rank. They used a fitness sharing/niching technique together with preference information in order to better cover the Pareto front and focus the selection procedure towards specific regions of the Pareto front. Their approach showed a similar or even better performance in some aspects than conventional techniques for non-linear system identification.

Instead of starting GP evolution with a random population, Langdon and Nordin [8] constructed if-then-else seeds from perfect individuals and used evolution under parsimony pressure as a way to enhance or keep good performance while reducing the size of individuals to attain good generalisation. Parsimony pressure is achieved by Pareto tournament selection where individuals can win either by being good or by being small. In this way bulky if-then-else clauses were replaced by elegant, short and general expressions. The technique is demonstrated with programmatic image compression, two machine learning benchmark problems and an insurance customer profiling task.

Journal of Theoretical and Applied Information Technology
20th May 2013. Vol. 51 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Ekàrt and Nèmeth [9] used MOGP to control tree bloat. They introduced a tournament-based selection as a variant of the Pareto non-domination criterion. This method selects solutions that are fitter and not much larger than the members of a tournament comparison set using a pre-defined parameter. They tried this method on several symbolic regression problems and Boolean multiplexer problems.

Bleuler *et al.* [10] suggested multi-objective optimisation for evolving compact GP trees by adding the size of the tree as a second objective. They compared SPEA2 to other bloat-reduction methods: Standard GP with tree depth limitation, constant parsimony pressure by adding a tree size dependent term to the fitness function, adaptive parsimony pressure by varying the complexity term according to the performance of the best individual found in the current generation, and two stage ranking method where priority is given to the main problem objective and then afterwards the program size. Results showed that the multi-objective approach maintained a lower average tree size compared to other methods and evolved more compact solutions slightly faster but without the need to specify parameters.

Zhang and Rockett [4]-[5] applied multi-objective genetic programming (MOGP) to search the feature space under the selective pressure defined by multiple objectives driven by the Pareto optimality concept. First, they mapped the input feature space into a one-dimensional decision space. Then they used Golden Section search as a fast means of finding the optimal threshold of a classifier for binary classification problems. They used three approaches to transform multiple class problems into a series of binary classification problems: One-from-*n*, single feature space mapping (SFSM) and hierarchical feature extraction (HFE).

2 METHODOLOGY

In this work, an optimised feature-extraction wrapper method is used by employing an evolutionary process as a search strategy to navigate in feature-subset space and there by construct mathematical transformations which map the input pattern space into a decision space which has maximum discrimination between the output classes figure2. To carry-out this search it is necessary to specify: i) A starting point, as well as a strategy to explore the space of the feature subsets, ii) an evaluation function and iv) a stopping criterion.

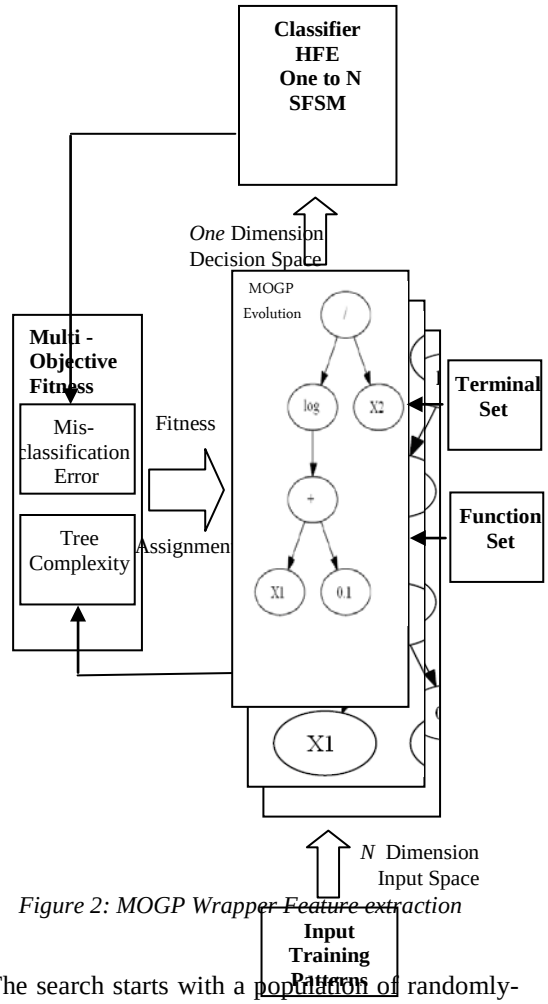

Figure 2: MOGP Wrapper Feature extraction

The search starts with a population of randomly-created GP trees. Each GP tree represents a sequence of mathematical operations which maps the input patterns into a one-dimensional decision space which is the output of the tree's root node. Then one of multi classification algorithms (One from N, HFE, SFSM) is used to find the optimum separates between classes. The output of the classifier is used to evaluate the performance of the feature extraction stage within the wrapper framework.

- Starting point is reached by creating the initial population of GP trees that represents feature extractors. Initial population is created randomly using the pre-defined terminal/function set and randomly selected input features to be as leaves of the tree.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

- The training stage is conducted repeatedly as follows.
  - The input training patterns are applied to the input of the trees in the GP population. Each tree is evaluated from bottom to top until the root node reached. The output of the root node represents a one-dimensional decision space.
  - The multi classifier algorithms uses the output from the GP individuals to perform the classification task, hence evaluating the 0/1 loss, the first component of each GP individual's vector of objectives.
  - A complexity measurement for each individual (the number of tree nodes) is used as the other component of an individual's fitness vector.
  - The evolutionary search for the optimum individuals is conducted by applying the evolutionary concepts of: selection, recombination and mutation.
  - A pre-defined criterion is examined to stop the search process when the final population represents a set of high-quality feature extractors.
- In the test stage, test patterns are mapped from the input space to the decision space using the evolved feature extractors from the MOGP Pareto front. The trained multi classifier is applied to classify test patterns according to their extracted features
- Given the above, the next step is to a select multi-objective genetic programming strategy to implement the evolutionary process with two goals [11]: To find a set of solutions as close as possible to the Pareto-optimal front as well as being as diverse as possible. A lot of research has been conducted in multi-objective genetic algorithms which can be extended to multi-objective genetic programming..
- In HFE and ONE to N approaches use policy to determined optimal threshold value to decide the final class label.

3. MULTI-CLASSFICATION ALGORITHMS

3.1 One-From-N Approaches

The most common decomposition techniques is one-from-*n* partitioning where for each class the feature extraction process is done to separate one class from all the other classes by combining all the others into a single pseudo-class. This process is repeated for each class which means we need *k* feature extractors for a *k* class problem. For one-from-*n* binary decompositions we also need a policy to deal with patterns which are multiply labelled to more than one class. A policy is also required to deal with unlabeled patterns during the all stages. For each model from the *k* models, all the points in the training data are used to build a feature extractor for each class. Then if the number of patterns for each class is *n* the total complexity of training is $O(nk^2)$.

The policy is using the threshold at each of the classification stages is applied in the projected one-dimensional decision space where the optimal threshold value is determined by minimising the misclassification error over the training set using golden section search. We use Distance between patterns and the mult-labelled patterns to decide the final class label.

3.2 HFE Approach

In hierarchical feature extraction, the partitioning of the problem is achieved by splitting it into subsets and then further splitting these subsets until no more splitting is possible. The choice of class hierarchy plays an important role in the accuracy of the hierarchical decomposition. This presents a problem when the number of classes is large, as the possible number of permutations of the hierarchy is very large. If the hierarchy is balanced we should obtain a hierarchy tree with $\log_2 k$ levels of binary decisions. The commonest approach to hierarchical feature extraction is to separate a given class against all the other (*k-1)* classes which are combined into one class table 1. For the next step another class is selected from the set of (*k-1)* classes to be separated against the remaining (*k-2)* classes; patterns of the first class are removed from this step. This process is continued repeatedly until only two classes remain which are then separated. A final stage is required to separate the last class from all other classes' patterns that were labelled in error by the preceding stages and so deal with the

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

'leakage' of these patterns down the classificational chain. Again a policy is needed to classify any unlabeled patterns that remain at the end. To calculate the total number of patterns used for training we will assume for simplicity that the hierarchy is balanced so at each level from top to bottom of the hierarchy tree the number of points used in training is halved. It was shown in [12] that the complexity of hierarchal feature extraction is $O(nk\log_2 k)$.

The policy is using the threshold at each of the classification stages is applied in the projected one-dimensional decision space where the optimal threshold value is determined by minimising the misclassification error over the training set using golden section search. We use Distance between patterns and the mult-labelled patterns to decide the final class label.

Table 1: HFE Approach Class Hierarchy

	Class 1	***Class 2***
Stage 1	Class1	Class2 , Class3 ,Class4…. , Class n
Stage 2	Class2	Class3, Class4 …. , Class n
Stage 3	Class3	Class4 , Class n
…	…	…
Stage n-1	Class n-1	Class n
Stage n	Class n	Class1 ,Clas23 …. ,Class n-1

3.3 Single Feature Space Mapping

Another approach to dealing with multi-class datasets without decomposing the problem is the single feature-space mapping (SFSM) which extracts features from the input space by mapping them to a new decision space with better discrimination between classes figure3. This new decision space a single dimension feature vector. A suitable classifier should be used following the feature extraction stage using either static/pre-defined class boundaries or dynamic class boundaries determined through the training process. The SFSM approach requires only one model and uses the training sample just once offering low computational cost relative to the one-from-*n* and pairwise approaches. In addition, it does not have the problem of in hierarchal feature extraction of defining an optimal class ordering. The three approaches to multi-class decomposition are shown in Table 2.

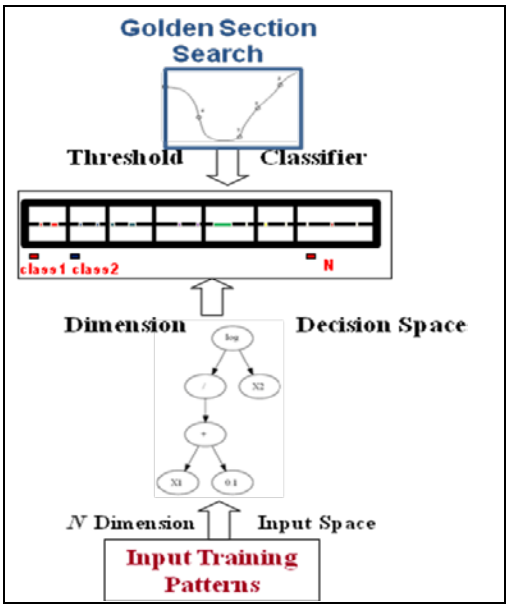

Figure 3: MOGP with The Single Feature-Space Mapping

Table2 :Comparison Between Multi-Category Classification Techniques

	One-of-*n*	Hierarchical	Pairwise	SFSM
No of Trained Models	*k*	*k-1*	*k (k-1) /2*	1
Training samples	*n k(k)*	$n\,k\,\log_2 k$	*n k (k-1)*	*n k*

4. THE UCI DATASETS

The datasets used in the current work are again from UCI Machine Learning database [13], the details of the datasets used can be referred to the table3 (Datasets): Glass (GLASS); Image segmentation (SEG); Teaching Assistant Evaluation (TAE); Thyroid (THY); Thyroid Gland (TGD); and Wine recognition (WIN). The MOGPs in this section are implemented using PCGP strategy.

Table 3: UCI machine Learning Database

Name	Features	Size	Classes

Journal of Theoretical and Applied Information Technology
20th May 2013. Vol. 51 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

IRIS	4	150	3
Thyroid Gland (TGD)	5	215	3
Wine recognition (WIN)	13	179	3
Teaching Assistant Evaluation (TAE)	5	151	3
Thyroid (THY)	21	7200	3

Hence cross-validation is commonly used to obtain a more accurate estimate of the empirical risk, where the available data set is partitioned into *5 cv2* parts and then *5 cv2* individual training-and-test processes are carried out. The average test risk now becomes the estimate of model generalization performance. Sometimes it is important to analyze the variance to provide a statistically significant measure at a certain confidence level.

The statistically measure used to compare classifier performance [14] where the *F*-measure was calculated for *TCR* values with different μ values to decide whether or not to reject the null hypothesis that the performances of the two spam filters were identical. Throughout this work we used a 95% confidence level to infer a statistical difference which is equivalent to an *F*- measure $\geq$ 4.74.

5. RESULTS AND ANAYLSIS

For the five datasets UCI shown in Table 4, MOGP (Single step) has selected in generating the best misclassification error, it can be seen that although the MOGP (Single step) method can deliver good results in many cases, it produces the (statistically) worst error rates of all classifiers for the WINE datasets.

Table 4: Mean Error Comparisons of Classifiers on Each Dataset

	MOGP		
dataset	ONE-N	HFE	Single S
TAE	0.471404	0.479351	**0.440404**
SEG	0.04108	0.035312	**0.02234**
IRIS	0.029333	0.025333	**0.02**
WINE	0.025996	**0.018231**	0.054621
THY	0.042139	0.035028	**0.024722**
TGD	0.05469	0.069948	**0.035352**

The MOGP (HFE) method shows that performs better in the WINE dataset and most of them are still worse than the MOGP (Single step) method.

Table 5:F-Statistic Comparisons of Classifiers on Each Dataset

Dataset	Classifiers		
	N_H	N_S	H_S
WINE	8.25425	5.56709	6.29514
THY	1.14617	1.07998	1.66056
TGD	1.00473	0.91275	14.9644
TAE	0.64974	1.98347	2.79374
SEG	1.11225	4.57854	2.66342
IRIS	2.11111	2.59989	0.99999

The comparisons in Table 5 show that *F*-tests were conducted between three classifiers of MOGPs. Apart from the comparison between three classifiers MOGPs over WINE dataset shows difference between the mean errors is not statistically significant, MOGP (Single step) performs shows there is statistically significantly difference with other classifiers over the five datasets. From table 6-11 shows confusion matrix each dataset over three approaches.

Table 6:Confusion matrix on IRIS dataset

N_IRIS				H_IRIS				SS_IRIS			
A / P	class1	class2	class3	A / P	class1	class2	class3	A / P	class1	class2	class3
class1	25	0	0	class1	25	0	0	class1	25	0	0
class2	0	24	1	class2	0	25	0	class2	0	25	0
class3	0	0	25	class3	1	1	23	class3	0	1	24

Table 7:Confusion matrix on SEG dataset

N_SEG				H_SEG				SS_SEG			
A / O	class1	class2	class3	A / P	class1	class2	class3	A / P	class1	class2	class3
class1	55	23	8	class1	69	13	0	class1	62	12	8
class2	0	147	24	class2	0	183	6	class2	21	164	4
class3	8	28	3338	class3	30	23	3219	class3	8	17	3247

Table 8:Confusion matrix on THY dataset

N_THY				H_THY				SS_THY			
A / P	class1	class2	class3	A / P	class1	class2	class3	A / P	class1	class2	class3
class1	58	19	6	class1	54	29	0	class1	59	18	6
class2	2	176	6	class2	3	179	2	class2	11	172	1
class3	4	29	3300	class3	15	28	3290	class3	4	18	3311

Table 9:Confusion matrix on TAE dataset

N_TAE				H_TAE				SS_TAE			
A / P	class1	class2	class3	A / P	class1	class2	class3	A / P	class1	class2	class3
class1	20	0	3	class1	18	4	1	class1	14	13	2
class2	10	6	9	class2	7	16	2	class2	4	18	3
class3	4	2	21	class3	4	9	14	class3	1	12	14

Table 10:Confusion matrix on TGD dataset

Journal of Theoretical and Applied Information Technology
20th May 2013. Vol. 51 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

N_TGD				H_TGD				SS_TGD			
A / P	class1	class2	class3	A / P	class1	class2	class3	A / P	class1	class2	class3
class1	84	0	0	class1	79	0	1	class1	79	1	0
class2	0	14	1	class2	0	16	0	class2	0	16	0
class3	1	0	16	class3	0	0	14	class3	0	0	14

Table 11:Confusion matrix on WINE dataset

N_WINE				H_WINE				SS_WINE			
A / P	class1	class2	class3	A / P	class1	class2	class3	A / P	class1	class2	class3
class1	31	0	0	class1	31	0	0	class1	31	1	0
class2	0	42	0	class2	0	42	0	class2	2	33	0
class3	0	1	20	class3	0	1	20	class3	0	0	27

In Figs. 4 – 7 shows the trees which have been generated by the evolutionary algorithms over datasets. It's clear that show MOGP (HFE) method less in the number of nodes from the rest of the approachs.

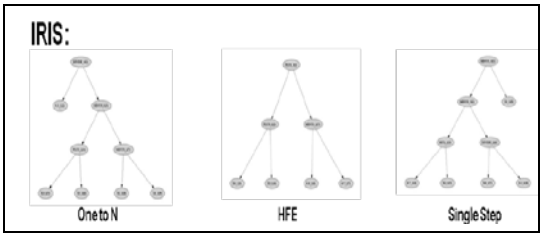

Figure 4:MMOGP transformation evolved for the IRIS dataset

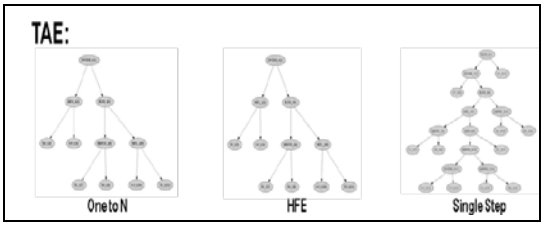

Figure 5:MMOGP transformation evolved for the TAE dataset

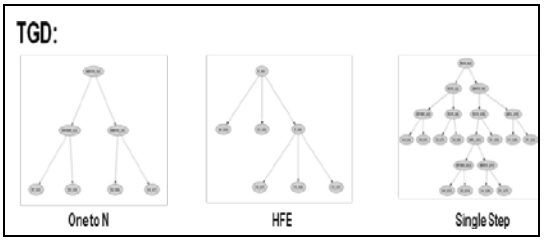

Figure 6:MMOGP transformation evolved for the TGD dataset

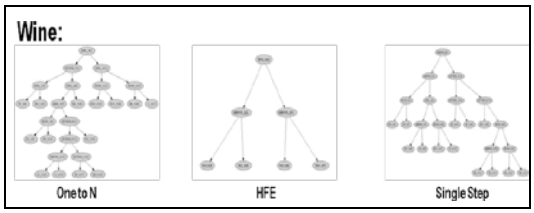

Figure7 :MMOGP transformation evolved for the Wine dataset

Those results are obtained by implement two objectives for each individual in the final population of 100 individual; misclassification error and the total number of nodes that constitute the individual trees. This result is after 20000 tree evaluations for each dataset.

6. CONCLUSIONS & FUTURE WORK

In this paper exploration for decision spaces of optimal dimensionality is performed with multi classification feature extraction. Three approaches within the MOGP framework were proposed to address the multiple class problems: one-from-*n*, single feature space mapping (SFSM), and hierarchical feature extraction (HFE). Statistical comparisons have been carried-out for the three approaches on six datasets containing multiple classes. On most of them, the three approaches introduced here give significantly better performance although (SFSM) is the best of the three on most datasets. the SFSM approach has advantages as it was fast, simple.

In future work The combination of different techniques allows us to make the most of several methods, leveraging on their strengths and avoiding their drawbacks[15].

GP can also evolve the setting for a neural network, an SVM, ANN classifier or any other conceivable classification mechanism. It is because of its flexibility that the same tool (GP) can be applied for different classification tasks, at a preprocessing or postprocessing [15].

REFRENCES:

[1] A. Bailey, "Class-Dependent Features and Multicategory Classification," Department of Electronics and Computer Science. PhD Thesis: University of Southampton, 2001.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[2] W.B. Langdon, *Genetic Programming and Data Structures: Genetic programming + Data Structures = Automatic Programming*. Kluwer Academic Pulisher, London, 1998.

[3] R. Poli, W. B. Langdon, and N. F. McPhee, A Field Guide to Genetic Programming: http://www.gp-field-guide.org.uk, 2008.

[4] Y. Zhang and P. I. Rockett, "Domain-Independent Approaches to Optimize Feature Extraction for Multi-Classification Using Multi-objective Genetic Programming," Department of Electronic and Electrical Engineering, University of Sheffield, Sheffield, UK Technical Report No. VIE 2007/001,2007.

[5] Y. Zhang and P. I. Rockett, "A Generic Optimising Feature Extraction Method Using Multiobjective Genetic Programming," Department of Electronic and Electrical Engineering, University of Sheffield, Sheffield, UK Technical Report No. VIE 2006/001,2006.

[6] K. Rodríguez-Vázquez, C. M. Fonseca, and P. J. Fleming, "Multiobjective Genetic Programming: A Nonlinear System Identification Application," in Late Breaking Papers at the 1997 Genetic Programming Conference, J. R. Koza, Ed.: Stanford Bookstore, 1997, pp. 207-212.

[7] K. Rodríguez-Vázquez, C. M. Fonseca, and P. J. Fleming, "Identifying the Structure of Nonlinear Dynamic Systems Using Multiobjective Genetic Programming," *IEEE Transactions on Systems, Man & Cybernetics - Part A: Systems & Humans*, vol. 34, no. 4, 2004, pp. 531-545.

[8] W. B. Langdon and J. P. Nordin, "Seeding Genetic Programming Populations," 3rd European Conference on Genetic Programming, 2000, pp. 304-315

[9] A. Ekárt and S. Z. Németh, "Selection Based on the Pareto Nondomination Criterion for Controlling Code Growth in Genetic Programming," *Genetic Programming and Evolvable Machines*, vol. 2, no. 1, 2001, pp. 61-73.

[10] S. Bleuler, M. Brack, L. Theile, and E. Zitzler, "Multiobjective Genetic Programming: Reducing Bloat Using SPEA2 " 2001 Congress on Evolutionary Computation, Seoul, Korea, 2001, pp. 536-543.

[11] K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms: John Wiley & Sons, Inc., 2001.

[12] A. Bailey, "Class-Dependent Features and Multicategory Classification," Department of Electronics and Computer Science. PhD Thesis: University of Southampton, 2001.

[13] C.L. Blake, and C.J. Merz , UCI Repository of machine learning databases. Irvine, CA: University of California, Department of Information and Computer Science (1998). [http://www.ics.uci.edu/~mlearn/MLRepository.html].

[14] K. Badran,"Multi-Objective Programming with an Application to Intrusion Detection in Computer," PHD thesis , University of Sheffield, June 2009.

[15] H. Jabeen and A. Baig, "A Framework for Optimization of Generic Programming Classifier Expressions for Binary Classification Using Particle Swarm Optimization," International Journal of Innovative Computing, Information and Control ICIC International, ISSN 1349-4198 Volume 8 , Number 1(A), January 2012.