Particle Swarm Optimization Based Tuning of Genetic Programming Evolved Classifier Expressions

Hajira Jabeen and Abdul Rauf Baig

Abstract. Genetic Programming (GP) has recently emerged as an effective technique for classifier evolution. One specific type of GP classifiers is arithmetic classifier expression trees. In this paper we propose a novel method of tuning these arithmetic classifiers using Particle Swarm Optimization (PSO) technique. A set of weights are introduced into the bottom layer of evolved GP classifier expression tree, associated with each terminal node. These weights are initialized with random values and optimized using PSO. The proposed tuning method is found efficient in increasing performance of GP classifiers with lesser computational cost as compared to GP evolution for longer number of generations. We have conducted a series of experiments over datasets taken from UCI ML repository. Our proposed technique has been found successful in increasing the accuracy of classifiers in much lesser number of function evaluations.

1 Introduction

Data classification has received increasing interest as a consequence of tremendous increase in data generating abilities due to automation. The task of classification can be viewed as labeling unseen data based upon some knowledge extracted from data with known labels. Automated classification algorithms are required to handle the problem of knowledge discovery from large amounts of data. Evolutionary algorithms have been found efficient in solving classification problems due to their stochastic global search mechanism.

"Genetic programming is an evolutionary computation technique that automatically solves problems without requiring the user to know or specify the form or structure of the solution in advance" [1].These inductively learned solutions are efficient in learning hidden relationships among data and discriminate them in a concise mathematical manner. Since introduction of GP, various methods have been introduced to for data classification using GP. These solutions range from derivation of decision trees [2], evolution of classification rules [3] and generation of SQL queries [4]. A relatively new GP based classification method is numeric expression trees [5]. These mathematical expressions trees are evolved using GP as discriminating expression for a certain class using some arithmetic functions and variables defined the in the primitive set. The variables are usually the attributes present in training data and some constants.

Hajira Jabeen · Abdul Rauf Baig
National University of Computer and Emerging Sciences,
Department of Computer Science, Islamabad, Pakistan
e-mail: hajira.jabeen@nu.edu.pk, rauf.baig@nu.edu.pk

The expression trees evolved using GP can form arithmetic or logical expressions based upon the primitive set used. In either case these expressions output a single numeric value and this value must be translated into the class labels. In case of binary classification one can simply assign one class to positive values and other class to negatives output values. The challenge arises in the case of Multi-Class classification problems where a single output has to be mapped to more than two classes. For this case methods like Static Range Selection [5] and Dynamic Range Selection [6] have been proposed.

GP presents numerous advantages for classification purpose. GP offers flexible and complex search space to search during classifier evolution. The classifier representations differ in each run, so we can eventually get several different classifiers with same or slightly different accuracy [7]. Easy and fast interpretation of result is possible as only one expression is evaluated to obtain the result [13]. The dependencies inherent in the data can be inducted into the classifier without expert intervention [7]. The classifiers are data distribution free, and able to operate upon the data in its original form [7].

Apart from above mentioned benefits, GP also suffers from the following issues:- GP based classification requires long training time. The classifiers increase in their complexity if necessary measures for avoiding code growth are not taken into account. GP yields different results after each run, both in structure of solution and accuracy.

In this paper we present a novel method for tuning of evolved classifiers making them more efficient and accurate. Several datasets with varying classes and attributes have been used to support the effectiveness of proposed tuning algorithm. Next section gives an overview of classification methods that use GP and an introduction to Particle Swarm Optimization PSO used for tuning. Method section explains the GP Algorithm used for classification and PSO based tuning algorithm proposed in this paper. Results section presents the experimental results followed by conclusions in the end.

2 Literature Review

2.1 Classification Using Genetic Programming

GP's outstanding abilities for the task of classification have been recognized since its inception [14]. Lots of work has been done to solve the problem of classification using GP. The main reason of interest in GP for classification is its ability to represent and learn solutions of varying complexity. In this section we will discuss some of the major techniques used for classification using GP.

Alex Frietas [4] has introduced a GP based classification framework where SQL queries are encoded into the GP Grammar. These are named Tuple Set Descriptor (TSD). The fitness of an individual is the number of rows satisfying the TSD. The framework incorporates lazy learning, i.e. rule consequences are evaluated first and one with the higher fitness is assigned to the rule. The advantages of using SQL based encoding is faster and parallel execution of queries, scalability and privacy. Another method using GP Classifier Expressions (GPCE) for

Multi-Class classification is used by Kishore et al [7]. A 'c' class problem is decomposed into 'c' two class problems. And a GPCE is evolved to discriminate each class from other classes. They define Strength of Association and Heuristic rules to tackle the conflicts arising between classifiers of different classes. They have used incremental learning and interleaved data format for speedup in the learning process. The work done by Bozarczuk et al [8] discovers classification rules for chest pain diagnosis. There are 12 classes present in the dataset and 165 attributes and 138 examples only. They have also evolved GP system for each class separately i.e. 12 times and chose best member as representative rule for each class. The fitness function takes into account classification accuracy, sensitivity, specificity and rule simplicity. At the end predictive accuracy of rule set as a whole and that of individual rules is evaluated. Loveard et al [6] have evaluated five different methods for Multi-Class Classification using strongly typed GP. One is binary decomposition in which the given problem is decomposed into binary problem, where one class is named as desired class and all other classes present in the data are assigned to reject class, the process is repeated for each class. Other method is static range selection, where the real output of a GP tree is divided into class boundaries and classes are assigned to input data based upon the GP tree output. Third method is dynamic range Selection where a subset of data is used to dynamically determine the class boundaries, and rest of the training data is used to evolve the classifiers. Fourth method is class enumeration, where a new data type is introduced into the terminal set of the GP trees. And all trees return a class type which is enumeration of the classes present in the data. The last method used is evidence accumulation, in which each GP tree contains a vector data storage corresponding to each class and these values are updated using new terminal which adds values ranging from -1 to 1 to the vector position particular class. The highest value is declared as the class outcome of tree. The results show that the dynamic range selection method is better for both binary and Multi-Class classification. Loveard et al have proposed two methods for classification of data containing nominal attributes [9]. One method considers splitting of GP execution based upon the value of a nominal attribute (execution branching) and other is conversion of a nominal attribute to binary or continuous attribute. Both methods are found efficient for classification of data containing nominal attributes. An interesting approach for Multi-Class classification using GP has been proposed by Muni [10]. In which collaborative view of classifiers for all the classes is considered. A Multi-Tree representation for classifier is presented. A chromosome has as many trees as there are classes in the data and each tree represents acceptor for samples belonging to its own class and rejecter to samples belonging to other classes. For evolution of this type of classifiers modified crossover operator is proposed. A new notion for unfitness of trees for genetic operations is proposed. A method Oring is proposed to combine results of classifier to achieve better performance. Heuristic rules and weight based scheme are also used to cater for ambiguous conflicting situations. The classifiers are can also output a 'do not know' when confronted with unfamiliar exemplars. A method for addition of weights has been proposed in [11] for Multi-Class object classification. The method for classification is range selection and the gradient descent method is used for searching during the evolution of GP Classifiers. This methodology makes the system more complex but it offers increase in performance of Genetic Programs. Many others methods have also been proposed for data classification using GP.

GP is found a very efficient innovative technique to handle to problem of data classification. GP suffers from an inherent drawback of inefficient code growth (bloat) during evolution. This increases the program complexity during the evolution process without effective increase in fitness. This increase in complexity has to be tackled explicitly by placing a bound on the upper limit of tree depth or nodes of the tree.

Another issue with GP based classification is long training time, which increases many folds with the increase in tree sizes during evolution. In this paper we have proposed a method that eliminates the need of evolving GP for longer number of generations and optimizes the GP evolved intelligent structures using PSO. Next section discusses some basics of PSO algorithm used for optimization in our proposed technique.

2.2 Particle Swarm Optimization

Particle Swarm Optimization Algorithm is originally formulated by Kennedy and Eberhart in 1995 [12]. Although it is also classified as an evolutionary algorithm but it models the sociological principle of bird flocking behavior during flying. The algorithm usually operates upon set of real multidimensional points scattered in the search space. These points move with certain velocity in the search space, mimicking bird's flight in search of optimal solution. The velocity of a particle in a given iteration is a function of the velocity of the previous step, its best previous position in the search space and the global best position. This exhibits a behavior of flying towards better position keeping in view its own best position and exploiting the knowledge of global best particle. The algorithm has been compared with various evolutionary algorithms and found equally efficient. Following are the update equations for particles in standard PSO.

$$X_i = X_i + V_i \tag{1}$$

$$V_i = \omega V_i + C_0 rand(0,1) X_{lbest} - X_i + C_1 rand(0,1) X_{gbest} - X_i \tag{2}$$

Where X $_{\text{gbest}}$ is the global best or local best particle and X $_{\text{lbest}}$ is the personal best of each particle. The values C_0 and C_1 are problem specific constants.

The Equation (1) is used to update position of a particle and Equation (2) is used to update the velocity of a particle during the PSO evolution process.

3 Methodology

In this section we will explain the algorithm used for classification and our proposed PSO based tuning method. The algorithm used for classification has been proposed by Muni [10]. One of the specialties of this algorithm is its Multi-Tree representation that makes it possible to evolve classifiers for multiclass classification in a single GP run. The evolved trees are in the form of arithmetic expressions elaborating relationships among different attributes of data. Each tree outputs a real value for each data instance. The output of a chromosome is a vector of real values. The tree corresponding to class label is trained to output positive value. Other trees must output negative value for a valid result.

Next section discusses the proposed optimization method that can increase the efficiency of GP evolved classifier expressions.

3.1 Tuning of Classifier Expressions

As mentioned in the previous section the classifiers contain attributes as terminals of the tree and a few constants. We have associated weights to all the terminals present in a tree. Consider a simple tree $((A_1+A_2)/A_3$ where A_1, A_2, and A_3 are attribute 1, 2 and 3 respectively. This tree will become $[(A_1*W_1) + (A_2*W_2)] / (A_3*W_3)$, after weight addition, where W_1, W_2 and W_3 are weights associated to each terminal. As shown in Figure1. The weight chromosome for this tree will be $[W_1, W_2, W_3]$. If the number of terminal nodes present in a tree is *'n'*. The number of nodes added for the sake of optimization is equal to *'2n'*. . Let *c* be the classes in the data , and *t* be the terminals in each tree: Then the total number of nodes added to chromosome will be :-

$$\sum_{i=1}^{c} 2 * t_i \tag{3}$$

In case of multi tree representation, we add weights to each tree in the chromosome. Let c be the classes in the data, and *'t'* be the terminals in each tree. The weight vector will be :-

$$[W_{ij}] \text{ where } i=1{:}c \text{ and } j=1{:}t \tag{4}$$

Here each weight is associated to the node *j* of the tree *i*. and we are interested in finding optimal value of this weight for each attribute in order to increase the efficiency of classifiers. An important point to note here is that the classifier remains intact if the values of all the added weights are set to '1'. Let CH_0 be the original chromosome and CH_w be the weight added chromosome, then

$$CH_0 = CH_w \text{ if } \forall \ [Wij]=1 \tag{5}$$

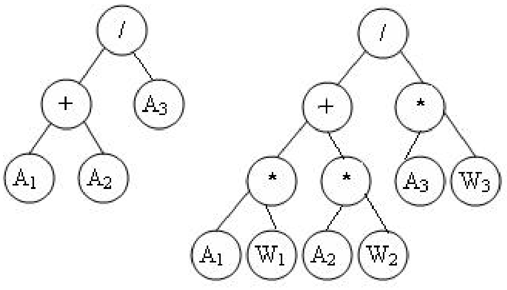

Fig. 1 Addition of weights for optimization

For the sake of optimization, a population of random particles having weights as their dimension is initialized. These weights are assigned random values between -1 and 1. This creates a multidimensional point in hyper space that has as many dimensions as there are weights in a GP chromosome corresponding to each terminal. PSO is used to evolve these weights for optimal values. The fitness of each particle is calculated by putting the values of weights in their corresponding positions and evaluating the accuracy of classifier for training data. We have used cognitive-social model that keeps track of its previous best as well as the global best particle. These weight particles are evolved for optimal position for a few generations until termination criteria is fulfilled.

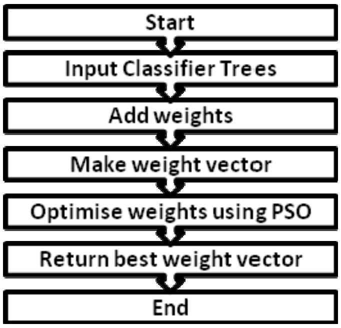

Fig. 2 Proposed tuning algorithm

4 Experimental Details

The data sets used for the classification purpose are taken from UCI repository. These data sets are Iris, Bupa, Wine, Glass and Wisconsin breast cancer. All these datasets are real valued data sets with varying number of classes and attributes. This is to prove the effectiveness of the proposed algorithm.

Each tree in the GP evolved classifier chromosome is appended by weights at its terminals, and the weights are evolved using PSO. The number of generations for evolution in GP is not kept fixed. The system is allowed to evolve until the fitness keeps on increasing. The evolution process is stopped only when the fitness increase is not observed for certain number of generations.

Table 1 GP parameters for Classification

S.No	Name	Value
1	Population size	600
2	Incremental Generations	20
3	Total generations	50
4	Maximum Depth	5

Table 1 lists the GP parameters used for the experimentation all the other parameters were kept same as mentioned in (10). Table 2 lists the parameters used for PSO. The results reported after tuning are averaged for 10 executions of PSO

Table 2 PSO parameters

S.No	Name	Value
1	No of particles	20
2	Initial value range	[+1 , -1]
3	Number of iterations	30

4.1 Iris Dataset

Iris data set is one of the simple and small data set used for classification. It has 4 attributes and three classes with 150 instances. Figure 3 shows increase in accuracy of GP classifiers before, and after tuning. It can be observed that PSO based

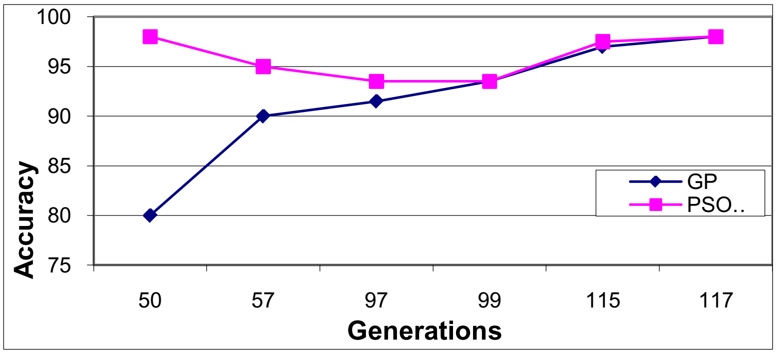

Fig. 3 Increase in Accuracy by PSO for Iris data

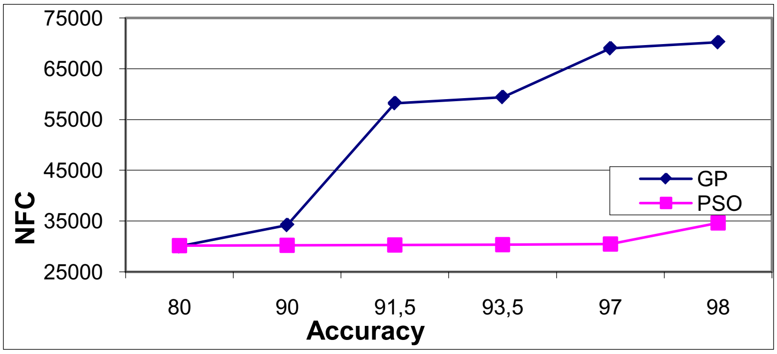

Fig. 4 NFC comparison with PSO for Iris data

tuning offers a considerable increase in most of the cases. Figure 4 compares the Number of Function calls used by GP and tuning method. GP achieves good results in much larger number of fitness evaluations as compared to PSO based tuning which achieves same accuracy in much lesser number of Function calls. PSO tuning method is efficient in finding better solutions in lesser number of Function evaluations. Results of 10 executions of PSO tuning on one classifier are reported in Table3 where PSO based tuning increases the accuracy in all the cases. Table 4 shows that on average 14% increase in accuracy is achieved.

4.2 Wisconsin Breast Cancer (WBC) Dataset

This data has two classes, 13 attributes and 699 instances. Figure 5 shows the increase in accuracy achieved for different GP classifiers. PSO tuning has increased the accuracy of simple GP classifiers. Figure 6 shows that PSO based tuning offers better accuracy with lesser number of functions calls(NFC) for achieving same accuracy as GP evolution process. Average increase in accuracy achieved is 7 % shown in Table 4. A result of 10 PSO runs is reported in Table3.

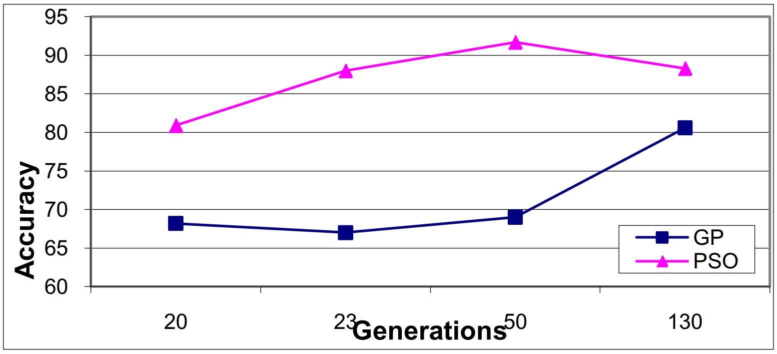

Fig. 5 Increase in Accuracy by PSO for Wbc data

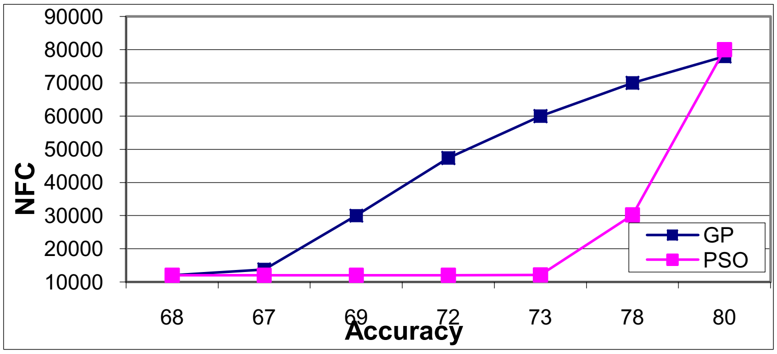

Fig. 6 NFC comparison with PSO for Wbc data

4.3 Glass Dataset

The Glass data has 6 classes and ten attributes having 214 instances. Figure 7 shows the increase in accuracy after tuning of GP classifiers. Figure 8 shows the difference in function calls to achieve same accuracy. PSO offered increase in the accuracy in much lesser number of function calls. Table 3 shows the result of 10 PSO runs and Table 4 summarizes the increase in accuracy achieved that was 1.7%.

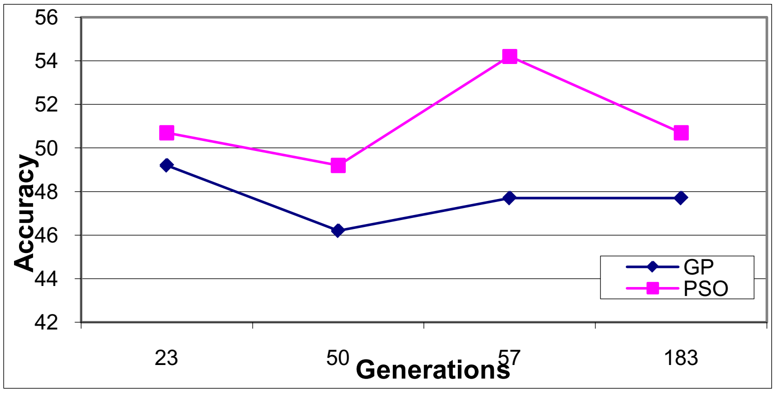

Fig. 7 Increase in Accuracy by PSO for Glass data

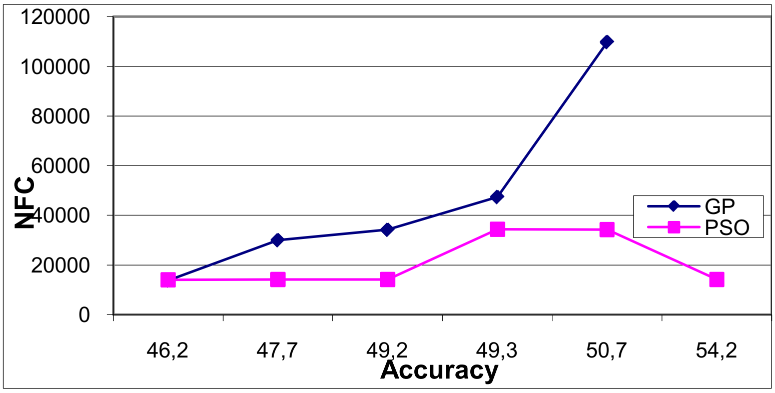

Fig. 8 NFC comparison with PSO for Glass data

4.4 Bupa Dataset

This dataset has 345 instances with 6 attributes and two classes. Figure 9 and Figure 10 show that tuning of weights has offered a prominent increase in the accuracy of the classifier with lesser number of function calls as compared to evolving GP for more number of generations. Average increase in accuracy achieved is

8.4% .Table3 presents results of 10 executions of PSO on one classifier evolved using GP. It is evident that in most of the cases weight tuning method offered an efficient increase in accuracy of the original classifiers. Table 4 gives an overview of increase in accuracy achieved.

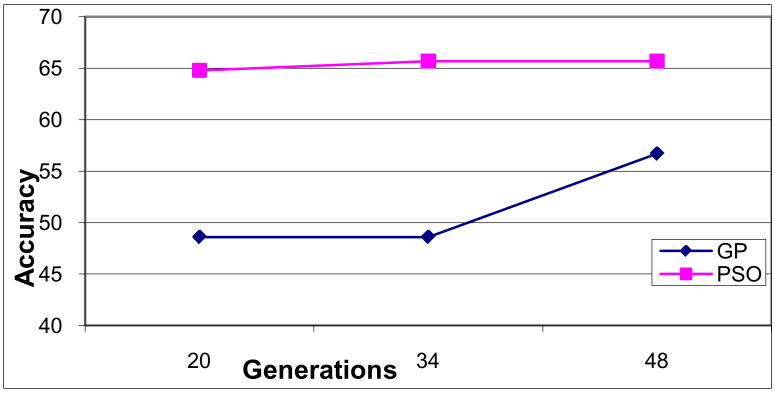

Fig. 9 Increase in Accuracy by PSO for Bupa data

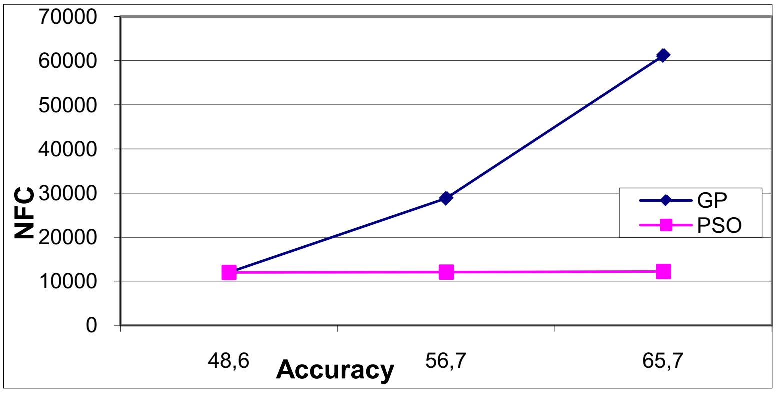

Fig. 10 NFC comparison with PSO for Bupa data

4.5 Wine Dataset

The Wine data set has 178 instances in 13 dimensions having three classes. Table 4 presents the result of 10 runs of PSO for the classifier evolved for Wine data. Figure 11 and Figure 12 show the increase in accuracy using the tuning method and difference in number of function calls in achieving the same accuracy. As observed in the previous cases, the PSO tuning method has achieved better accuracy with lesser function evaluations. The average increase achieved is 6.5% shown in Table 3.

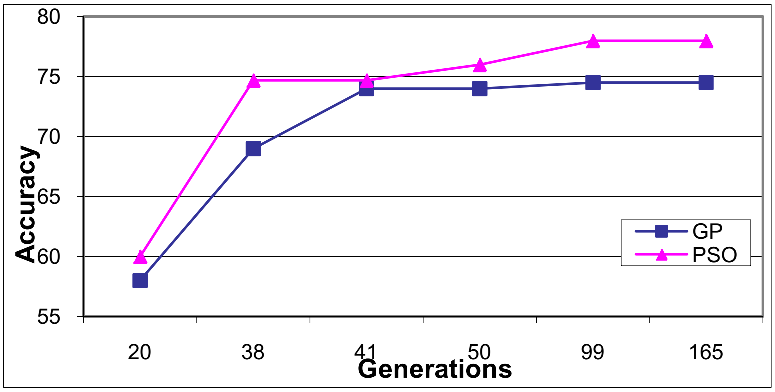

Fig. 11 Increase in Accuracy by PSO for Wine data

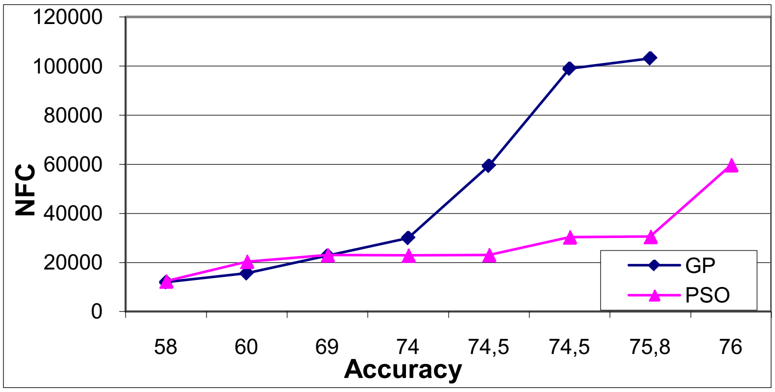

Fig. 12 NFC comparison with PSO for Wine data

Table 3 Increase in accuracy after 10 PSO runs

Datasets	IRIS	GLASS	BUPA	WINE	WBC
GP Accuracy	80.5%	46.0%	56.0%	70.0%	69.8%
PSO_1	98.0%	47.8%	65.7%	79.3%	78.4%
PSO_2	93.5%	49.2%	64.8%	79.3%	78.4%
PSO_3	95.5%	47.7%	65.7%	70.6%	74.7%
PSO_4	93.5%	47.7%	65.7%	70.6%	74.7%
PSO_5	91.0%	47.7%	65.7%	80.0%	78.0%
PSO_6	93.5%	47.8%	65.7%	80.0%	77.9%
PSO_7	95.5%	49.2%	64.8%	78.0%	79.0%
PSO_8	92.0%	47.8%	65.7%	79.3%	75.8%
PSO_9	96.0%	47.8%	64.8%	70.6%	74.1%
PSO_{10}	96.0%	49.2%	65.7%	78.0%	77.9%

Table 4 Average Increase in Accuracy

Datasets	Average Increase in Accuracy	Maximum Accuracy Achieved	Minimum Accuracy Achieved	Standard Deviation
IRIS	14 %	98.0%	91.0%	0.02
WBC	7.1%	78.4%	74.1%	0.01
GLASS	1.9%	49.2%	47.7%	0.006
WINE	6.5%	92.5%	87.5%	0.04
BUPA	8.4%	71.6%	65.0%	0.02

5 Conclusions

In this paper we have proposed a new method for the tuning of classifier expressions evolved by GP, it has been shown that this method tends to increase the training as well as testing accuracy of the classifiers. This method can eliminate the need for evolving GP classifiers for longer number of generations in search of better accuracy. It also helps in reducing the number of function evaluations desired for GP evolution. The more number of generations in GP also means increase in GP tree sizes over generation making the task more complex. On the other hand, in case of PSO based tuning we can get better results in much lesser number of function evaluations with increase in depth of trees by only one level. This increase in tree complexity gives an attractive outcome of increase in corresponding accuracy. Future work includes determination of optimal parameters for PSO for tuning and use of different variants of PSO for tuning.

References

1. Poli, R., Langdon, W.B., McPhee, N.F.: A Field Guide to Genetic Programming (2008)
2. Eggermont, J.: Data Mining using Genetic Programming: Classification and Symbolic Regression. Leiden University, PhD Thesis (2005)
3. Bojarczuk, C.C., Lopes, H.S., Freitas, A.A.: Discovering Comprehensible Classification Rules using Genetic Programming: A Case Study in a Medical Domain. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 953–958. Morgan Kaufmann, San Francisco (1999)
4. Freitas, A.A.: A Genetic Programming Framework For Two Data Mining Tasks: Classification And Generalized Rule Induction. In: Genetic Programming, pp. 96–101. Morgan Kaufmann, USA (1997)
5. Zhang, M., Ciesielski, V.: Genetic Programming For Multiple Class object Detection. In: Proceedings of the 12th Australian Joint Conference on Artificial Intelligence, Australia, pp. 180–192 (1999)
6. Loveard, T., Ciesielski, V.: Representing Classification Problems in Genetic Programming. In: IEEE Congress on Evolutionary Computation, pp. 1070–1077 (2001)

7. Kishore, J.K., et al.: Application of Genetic Programming for Multicategory Pattern Classification. IEEE Transactions on Eolutionary Computation (2000)
8. Bojarczuk, C.C., Lopes, H.S., Freitas, A.A.: Genetic programming for knowledge discovery in chest-pain diagnosis. IEEE Engineering in Medicine and Biology Magazine, 38–44 (2000)
9. Loveard, T., Ciesielski, V.: Employing nominal attributes in classification using genetic programming. In: 4th Aisa pacific conference on simulated evolution and learning, Singapore, pp. 487–491 (2002)
10. Muni, D.P., Pal, N.R., Das, J.: A Novel Approach To Design Classifiers Using GP. IEEE Transactions of Evolutionary Computation (2004)
11. Zhang, M., Smart, W.: Genetic Programming with Gradient Descent Search for Multiclass Object Classification. In: Keijzer, M., O'Reilly, U.-M., Lucas, S., Costa, E., Soule, T. (eds.) EuroGP 2004. LNCS, vol. 3003, pp. 399–408. Springer, Heidelberg (2004)
12. Kennedy, J., Eberhart, R.C.: Particle Swarm Optimization. In: IEEE International Conference on Neural Networks, pp. 1942–1948 (1995)
13. Engelbrecht, A.P., Schoeman, L., Rouwhorst, S.: A Building Block Approach to Genetic Programming for Rule Discovery, in Data Mining: A Heuristic Approach. [book auth.]. In: Abbass, H.A., Sarkar, R., Newton, C. (eds.) Data Mining, pp. 175–189. Idea Group Publishing, USA (2001)
14. Koza, J.R.: Genetic Programming: On the Programming of computers by Means of Natural Selection. MIT Press, Cambridge (1992)