International Journal of Innovative Computing, Information and Control
Volume **8**, Number **1**, January **2012**

ICIC International ©2012 ISSN 1349-4198
pp. 1–10-06097

GPSO: A FRAMEWORK FOR OPTIMIZATION OF GENETIC PROGRAMMING CLASSIFIER EXPRESSIONS FOR BINARY CLASSIFICATION USING PARTICLE SWARM OPTIMIZATION

Hajira Jabeen[1] and Abdul Rauf Baig[2]

[1]Iqra University
5, Khayaban e Johar, H-9/1, Islamabad, Pakistan
hajirajabeen@hotmail.com

[2]National University of Computer and Emerging Sciences, NU-FAST
A.K. Brohi Road, H-11/4, Islamabad, Pakistan
rauf.baig@nu.edu.pk

Received June 2010; revised January 2011

Abstract. *Genetic Programming (GP) is an emerging classification tool known for its flexibility, robustness and lucidity. However, GP suffers from a few limitations like long training time, bloat and lack of convergence. In this paper, we have proposed a hybrid technique that overcomes these drawbacks by improving the performance of GP evolved classifiers using Particle Swarm Optimization (PSO). This hybrid classification technique is a two-step process. In the first phase, we have used GP for evolution of arithmetic classifier expressions (ACE). In the second phase, we add weights to these expressions and optimize them using PSO. We have compared the performance of proposed framework (GPSO) with the GP classification technique over twelve benchmark data sets. The results conclude that the proposed optimization strategy outperforms GP with respect to classification accuracy and less computation.*
Keywords: Genetic programming, Classification, Particle swarm optimization, Optimization, Expressions

1. **Introduction.** The ideal classifiers help in future decision making and reveal hidden relationships in the data. However, finding the ideal classifier is a difficult task. The global search mechanisms like evolutionary algorithms have been found efficient for classification problems. Genetic Programming (GP) is one of the evolutionary algorithms found successful for classification. GP can autonomously search for classifiers with flexible structure and size. These classifiers can learn the data relationships during evolution. The resulting classifiers are comprehensive and portable to tools like spread sheets. GP is considered a powerful classification tool owing to its characteristics like flexibility, representation of variable length solutions, ability to model the data relationships, comprehensiveness and portability. However, GP suffers from a few well known limitations like long training time (every member of the population is a classifier), code bloat (the size of classifier population starts growing) and lack of convergence (one may not always get the optimal classifier). In this paper, we present a novel method for optimization of GP evolved classifiers using Particle Swarm Optimization (PSO). In this hybrid approach, (GP+PSO = GPSO), we have overcome a few limitations of GP in the following way.

1. Reduced bloat by using a specialized depth limited crossover operator (and early stopping).
2. Reduced training time by stopping the evolution process early (and reducing bloat and computation).

3. Reduced the computation (function evaluation) by optimizing the premature classifier using PSO.

We have tested the performance of our proposed GPSO hybrid framework over several binary classification data sets from UCI and found it efficient on all the classification problems.

2. **Related Work.** Although introduced earlier [1,2], GP was popularized in 1992 after Koza 痴 work [3]. GP allows variable and flexible representations of solutions, making it applicable for a wide variety of problems. GP can perform the data classification in several ways. One of the methods is to evolve classification algorithms through GP [4,5]. These methods involve defining a grammar to initialize genetic programs for evolution. The resultant algorithms include neural networks, decision trees, fuzzy rule based systems and fuzzy Petri nets. GP has been commonly used for evolution of classification rules by evolving logical rules of the form if-then-else [6,7]. A novel GP based classification technique is evolution of arithmetic classifier expressions (ACE) that serve as a discriminating function between classes. These expressions have real value as their output. Static threshold [8], dynamic threshold [9] or slotted threshold [10] is applied to this output for classification decision. Another scheme is binary decomposition method [11,12]. We have used ACE based classification due to efficient results and transparency.

Some researchers have performed a simplification step on GP expressions [13] or tree complexity penalty in fitness evaluation (parsimony pressure) [14], offspring size limits in crossover [15] or limits on maximum tree size [12]. A unique method [16,17] performs gradient search for optimization of terminals, in GP trees, for symbolic regression. However, these methods have not been introduced for classification tasks.

In this paper, our emphasis is to optimize ACE in less computation. For this purpose, we have investigated the use of Paricle Swarm Optimization Algorithm (PSO). It is computationally efficient and effective for a wide variety of applications like function optimization [18] or neural network training [19]. PSO solves optimization problems by simulating bird flock flying together in search of some ideal place. Considerable research is done for optimization and efficient working of PSO. Several parameters are introduced to improve the performance of PSO. Clerc and Kennedy [20] analyzed the convergence behavior of the PSO and introduced the constriction factor,which improves the exploration ability of swarm. Shi and Eberhart [21] introduced the concept of linearly decreasing inertia weight which determines the step size and direction for movement. Shi [22] proposed a fuzzy, nonlinear inertia updating strategy. Optimal values of constriction coefficients and inertia weights proposed by Clerc, reported by Poli in [23] are 0.7298 and $C_0 = C_1 = 1.49618$. To improve the performance of PSO, another research focus has been variations in PSO topology. Keneddy [24] proposed that PSO with smaller neighborhood performs better on complex problems and larger neighborhood would perform better on simpler problems. Suganthan [25] suggests a dynamically increasing neighborhood, until it includes all the particles of the swarm. Parsopoulos and Vrahatis used a combination of the global version and localised version to make a unified particle swarm optimizer (UPSO) [26]. Mendes and Kennedy introduce a fully informed PSO [27], 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. Various modifications to the PSO include, attractive-repulsive PSO (ARPSO), [28] predator-prey approach to PSO (PPO) [29] species based PSO (SPSO) [30] and charged swarm [31].

Despite proposition of various GP and PSO hybrid algorithms [32], we have not been able to find any optimization performed on GP evolved expressions using PSO.

3. **Proposed GPSO Framework.** The binary classifier is a many-to-one mapping that maps the input data to a class. It takes a feature vector as input and assigns a class label to it. In this paper, we are interested in finding an optimal arithmetic classifier expression. Such an expression outputs a real value which is translated into class label. A GP-ACE is evaluated for an input data, and if the output of the GP-ACE is $= 0$, the input data is assigned to one class; otherwise, it is assigned to the other class. The best classifier can classify the maximum of training samples, correctly. In this work, we optimize the performance of the GP evolved ACE by addition and tuning of weights assigned to all terminal values in the classifier. This optimization is performed using PSO. Figure 1 gives an overview of the proposed GPSO algorithm.

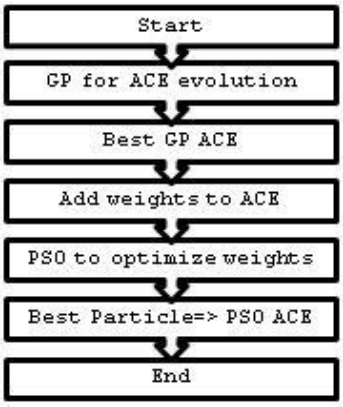

FIGURE 1. GPSO algorithm

The first step of proposed algorithm is to evolve ACE using GP. The best ACE is prepared for optimization by the addition of weights along each terminal. These weights form a PSO particle and optimized using PSO. The description of GPSO is given in the Algorithm GPSO.

Algorithm GPSO

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

The proposed framework involves three phases:

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

3.1. **GP phase.** The GP creates a program with the help of a function set and terminal set. The function set used for ACE evolution is $+$, $-$, $*$, $/$. The terminal set contains attributes of the data and random constant. The initialization method used for ACE, is the well known ramped half and half method [3].

Three operators, mutation, reproduction and crossover are used for ACE evolution. For mutation, a random candidate is selected, if it is a function node, it is replaced by a random function node; else, it is replaced by an arbitrary terminal node [11]. The reproduction operator selects a tree based on proportionate selection and copies that

tree into the next generation. The parents for crossover are selected by the tournament selection. The DepthLimited [33] crossover has been used in our approach, where two sub-trees (from parents) of same depths are swapped.

The classifier is trained to output a positive response (accept) for one class and negative response (reject) for the instances belonging to the other class. The training data is fed to a classifier and its classification accuracy is measured. We have used the two layered fitness as proposed by Frietas [12]. The classifier with better accuracy is always preferred, and if the accuracy of two classifiers is equal, then the simpler classifier is selected.

Algorithm Fitness

1. Begin
2. int count-correct = 0
3. For all instances in the training data N
4. Evaluate the classifier expression using the attribute values from the given instance (Value)
5. If Value >= 0 and class = desired class
6. count-correct++
7. if Value < 0 and class = not desired class
8. count-correct++
9. End if
10. End for
11. Fitness = (count-correct/N)∗100
12. End

The output of this phase is an ACE that is trained to differentiate between two classes.

3.2. **Preparation for optimization of classifiers.** An ACE has two types of terminals, attribute values $[A_1, \ldots, A_n]$ and ephemeral constant [0-10]. We assign unique weights to all the terminals present in ACE by the addition of a '$*$' node and a weight node. Let A_0 be an ACE and t be the number of terminals in the classifier then the weight vector will be:

$$[W_j], \text{ where } j = 1 : t \tag{1}$$

This process increases the complexity of the ACE and 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 affects 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:

$$T_{nj} = W_j * T_{oj}, \text{ where } j = 1 : t \tag{2}$$

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 An be the weight added ACE, then

$$A_0 = A_n, \text{ if } [W_j] = 1 \text{ where } j = 1 : t \tag{3}$$

Figure 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. These weights combine to form a weight vector $= [W_1, W_2, W_3]$. Similar weight vectors are formed for the best ACE evolved in the previous phase for the optimization.

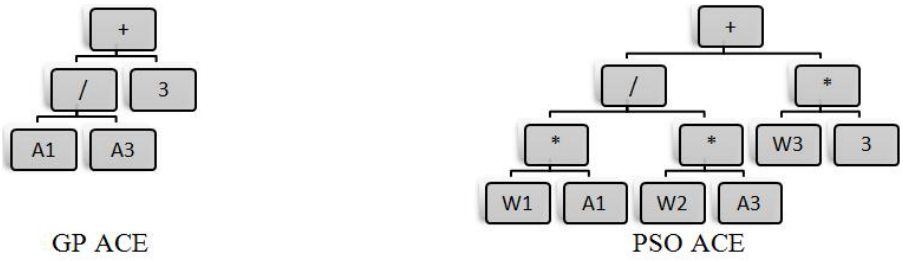

FIGURE 2. Addition of weights

3.3. **PSO phase for optimization of classifiers.** The important step in PSO is initialization of position and velocities of particles for optimization. The empirical analysis of our problem revealed that random weight initialization may not lead to optimal results. Whereas, we are interested in increasing the efficiency of classifiers. Therefore, we have incorporated a special initialization mechanism that favors better particles. It has been seen that efficient initialization leads towards better solutions and tends to find the optimum solutions in less effort.

We have used random weight initialization 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. We iterate this random initialization process certain number of times (P_{count}). After that, the best of P_{count} is returned as the swarm particle. The iteration process is stopped earlier if the solution with better or equal fitness is found. The velocities of particles are initialized randomly. After the initialization, the PSO algorithm is proceeded in the traditional way. The movement of the swarm particles is controlled by their velocity and position update equations. We have used the full model of PSO that uses social and cognitive components for movement. The velocity of particles is updated by the following equation:

$$V_{i+1} = wV_i + C_0 \text{ rand } (0,1)(X_{pbest} - X_i) + C_1 \text{ rand } (0,1)(X_{gbest} - X_i) \tag{4}$$

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 best of a particle. The constriction coefficients control the convergence of the particles. We have used the Clerc's constriction method which recommends following values:

$$C_1 = 1.49618, \quad C_2 = 1.49618 \quad \text{and} \quad w = 0.7298 \tag{5}$$

We have used the neighborhood model 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. Kennedy [36] suggested that the neighborhood of smaller size works better for complex problems. After velocity update, the particles update their positions using the following equation:

$$X_{i+1} = X_i + V_i \tag{6}$$

All the particles in the swarm 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. The condition is either a number of iterations, or certain threshold on fitness achieved by best particle. The optimization algorithm is explained in Algorithm PSO. This is last of the three phases that form the proposed hybrid combination of GP and PSO (GPSO) technique.

Algorithm PSO

1. Begin
2. Initialize particles using algorithm in Figure 3
3. While termination criteria not met
   (a) Calculate fitness of particles
   (b) For each particle
      (i) Update particle gbest
      (ii) Update particle velocity using Equation (4) and values from 5
      (iii) Update Particle Position using Equation (6)
   (c) End particles
4. End while
5. Return new population
6. End

4. **Experimental Settings.** The classification accuracy of GPSO has been compared with the simple GP based classification approach. We have used 12 benchmark binary classification problems from UCI ML repository. All the data used for classification is real valued except the BUPA data set which has a categorical attribute with numerical values. We have treated this attribute as a numerical attribute [24]. Similarly, Statlog (Heart) data set has few binary and nominal values, which were treated like numeric values. In case of WBC data, there are some missing values, which have been deleted.

The parameters used for classifier evolution using GP are mentioned in Table 1. These parameters have been carefully selected for classification after empirical analysis in our previous work [33].

Table 1. GP parameters

Parameters	Values
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 with Max depth 6
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

Table 2 presents the PSO parameters used for GP classifier tuning in GPSO approach. We have used neighborhood model suggested by Kennedy [24]. Other parameters like constriction coefficients and inertia weight have been adopted from Clerc's analysis [23]. These parameters have been widely used and found efficient for optimization tasks.

5. **Results.** The GP results have been generated using 10 fold cross validation, twice, with different initial populations. This process is repeated five times with a different tenfold partitioning and random sampling of the data. In this way, we have performed 10 GP runs (each involving tenfold cross validation). All the parameters in GP are kept same except the new initialization in each run [12,15]. In every single GP execution (total 100), we extracted a classifier after every 10 generations, after that, we performed PSO

TABLE 2. PSO parameters

Paramaeter	Values
Particles	100
Initial values	$[-1, 1]$
Dimensions	Number of leaves
Iterations	30
C_0	1.49 [23]
C_1	1.49 [23]
W	0.7 [23]
Model	Lbest model [23]
Neighborhood size	2
P_{count}	10

optimization ten times for every classifier. The classification accuracy and number of function evaluations are observed for every classifier for GP and PSO. These results are averaged for all the GP executions (100) and PSO executions (1000). These averages for the test data have been reported in the result. We have used classification accuracy and the number of function evaluations (NFE) performed to achieve that accuracy, as performance metrics. These measures represent better performance and less computation. Figure 3 presents a graphical representation of the increase in accuracy achieved after optimization process for WBC data after every ten generations. Where GP represents the accuracy of classifier obtained by GP only and GPSO is the classifier obtained after the optimization process. This can be seen that the optimization process has successfully increased the accuracy of a classifier from 95.5% to 97.2% after ten generations while the accuracy achieved by GP remained less than 97% until 120 generations. On the other hand, the GPSO improved the classification accuracy up to 99%.

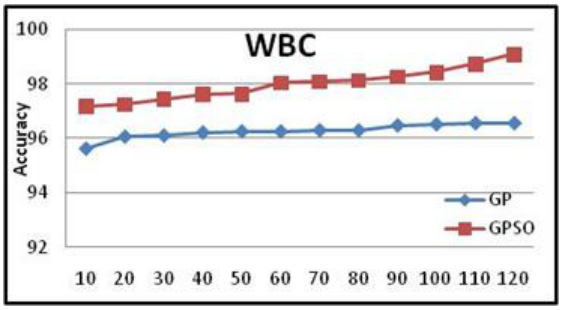

FIGURE 3. Average increase in testing accuracy after optimization

Table 3 presents the classification results of GPSO and traditional GP classification in testing phase. We have extracted a classifier after every 10 generations and analyzed the effect of optimization over various stages of evolution. The results presented in these tables are average of 100 GP runs and 1000 PSO runs, as discussed previously. The row GP-NFE tells the number of function evaluations elapsed to achieve given result, these NFE increase with the increase in generations. The row GPSO-NFE tells the number of function evaluations performed to achieve the results presented in the given column. This can be seen that the GPSO has achieved better classification accuracy and outperformed traditional GP in less number of function evaluations (NFE) (compare the results across the columns). We have experimented with twelve data sets. This can be seen that the GPSO results are better than the GP phase alone. Another point worth mentioning is

that GPSO improves the accuracy of classifiers many folds in less function evaluations in most of the cases. On the other hand, GP alone has not achieved compatible accuracy after many generations.

TABLE 3. Comparison of accuracies using GP and GPSO

Datasets	Generations	10	20	30	40	50	60	70	80	90	100	110	120
GP-NFE∗104		97.2	97.3	97.5	97.6	97.6	98.0	98.1	98.1	98.3	98.4	98.7	99.1
WBC	GP %	95.6	96.1	96.1	96.2	96.3	96.3	96.3	96.3	96.5	96.5	96.6	96.6
	GPSO %	97.2	97.9	98.1	98.7	99.0	99.0	99.1	99.1	99.1	99.1	99.1	99.6
	GPSO NFE ∗ 10 4	0.9	1.5	2.1	2.8	3.4	4.0	4.5	5.1	5.8	6.4	6.9	7.5
BUPA	GP %	64.7	66.1	68.5	69.3	68.4	69.0	68.2	68.5	68.7	69.1	69.3	69.0
	GPSO %	68.7	69.0	69.6	70.7	70.9	70.0	70.5	70.4	70.3	71.0	71.1	70.9
	GPSO NFE ∗ 10 4	0.9	1.6	2.2	2.7	3.4	3.9	4.5	5.2	5.7	6.3	6.9	7.6
HABER	GP %	73.2	72.9	72.3	72.5	71.8	72.2	72.0	71.5	71.6	72.5	72.5	72.5
	GPSO %	74.8	74.4	73.4	74.1	74.1	73.6	74.1	73.6	73.4	74.4	74.1	74.3
	GPSO NFE ∗ 10 4	0.9	1.5	2.1	2.8	3.4	4.0	4.6	5.1	5.7	6.3	7.0	7.6
PARK	GP %	76.6	79.6	79.8	80.4	79.3	81.0	79.8	80.7	80.8	81.2	80.4	80.4
	GPSO %	80.4	81.8	83.0	83.5	80.8	83.9	82.7	82.4	82.5	83.9	83.1	85.0
	GPSO NFE ∗ 10 4	1.0	1.6	2.2	2.8	3.4	4.0	4.6	5.2	5.8	6.4	7.0	7.5
PIMA	GP %	65.7	65.4	65.3	66.8	66.9	66.1	67.7	67.8	68.2	68.3	68.5	68.6
	GPSO %	71.3	71.6	70.9	71.5	70.8	70.7	73.6	73.6	74.0	74.1	74.7	74.4
	GPSO NFE ∗ 10 4	1.0	1.5	2.1	2.8	3.4	4.9	4.6	5.2	5.7	6.3	7.0	7.6
TRANS	GP %	75.8	76.4	76.7	76.7	77.1	77.1	77.3	77.2	77.3	77.2	77.2	77.4
	GPSO %	76.6	77.4	77.8	77.6	78.0	77.8	77.7	77.8	77.6	77.8	77.5	78.1
	GPSO NFE ∗ 10 4	1.0	1.6	2.1	2.7	3.4	4.0	4.6	5.2	5.8	6.4	7.0	7.6
ION	GP %	78.5	83.2	84.8	85.6	86.1	86.3	86.6	86.9	87.2	87.6	88.4	88.5
	GPSO %	82.1	84.1	86.4	87.0	87.4	87.6	88.0	88.4	88.8	89.0	89.1	89.3
	GPSO NFE ∗ 10 4	1.0	1.6	2.1	2.7	3.4	4.0	4.6	5.1	5.8	6.4	7.0	7.6
SPEC	GP %	79.4	79.0	79.1	78.6	79.1	78.9	78.9	79.0	78.6	78.8	78.8	77.6
	GPSO %	80.9	79.7	80.1	79.6	80.1	80.1	80.2	79.5	79.0	79.3	80.1	78.5
	GPSO NFE ∗ 10 4	1.0	1.6	2.2	2.7	3.3	4.0	4.6	5.2	5.8	6.4	7.0	7.6
RIPPER	GP %	87.4	87.5	88.3	88.7	88.5	88.6	88.7	88.8	88.6	88.9	89.1	89.1
	GPSO %	87.5	88.0	88.6	89.1	88.8	89.6	89.3	89.2	89.1	89.5	89.5	89.5
	GPSO NFE ∗ 10 4	1.0	1.6	2.2	2.7	3.3	4.0	4.6	5.2	5.8	6.4	7.0	7.6
SONAR	GP %	70.4	73.8	72.6	73.6	72.7	73.5	73.6	73.2	72.7	73.6	73.5	73.3
	GPSO %	70.4	74.0	72.7	74.1	73.4	74.1	74.1	74.0	73.6	74.4	74.0	73.9
	GPSO NFE ∗ 10 4	1.0	1.6	2.1	2.7	3.3	4.0	4.6	5.2	5.8	6.4	7.0	7.6
MUSK	GP %	62.3	64.0	64.3	65.6	66.5	66.8	67.3	67.6	67.9	68.2	68.5	69.3
	GPSO %	65.1	67.9	68.3	69.2	69.6	70.0	70.3	70.5	70.8	71.2	72.1	73.4
	GPSO NFE ∗ 10 4	1.0	1.6	2.1	2.8	3.4	4.0	4.6	5.1	5.7	6.4	6.9	7.5
HEART	GP %	73.4	77.5	78.3	80.4	82.3	82.3	83.4	88.3	88.3	88.3	88.3	88.3
	GPSO %	75.6	79.6	81.6	82.7	84.0	84.7	85.4	88.3	90.7	90.8	91.1	91.7
	GPSO NFE ∗ 10 4	0.9	1.5	2.1	2.7	3.4	3.9	4.5	5.1	5.7	6.3	6.9	7.5

6. **Conclusions.** In this paper, we have presented a novel hybrid method for efficient tuning of GP evolved ACE. This method has proved that the accuracy of arithmetic classifier can be increased by addition and optimization of weights associated with the terminals. We can extract an intelligent structure of ACE in a few generations of GP and increase its performance by addition and optimization of weights. This method has outperformed other GP based classification approaches. However, the proposed technique is applicable to real valued data only. Mixed attribute type data or categorical data must be converted into numerical values for this approach. The optimization process can be applied to arithmetic expressions only; the use of this technique for rule-based classifier can be explored. Some other future research directions include extension of proposed

algorithm for multi-class classification and use of proposed tuning algorithm in other variants of GP used for classification.

REFERENCES

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

[2] R. P. Salustowicz and J. Schmidhuber, Probabilistic incremental program evolution, *Evolutionary Computation*, pp.123-141, 1987.

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

[4] G. A. Pappa and A. A. Freitas, Evolving rule induction algorithms with multiobjective grammer based genetic programming, *Knowledge and Information Systems*, 2008.

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

[6] A. P. Engelbrecht, L. Schoeman and S. Rouwhorst, A building block approach to genetic programming for rule discovery, *Data Mining: A Heuristic Approach*, pp.175-189, 2002.

[7] I. D. Falco, A. D. Cioppa and E Tarantino, Discovering interesting classification rules with GP, *Applied Soft Computing*, pp.257-269, 2002.

[8] M. Zhang and V. Ciesielski, Genetic programming for multiple class object detection, *Proc. of the 12th Australian Joint Conference on Artificial Intelligence*, Australia, pp.180-192, 1999.

[9] W. R. Smart and M. Zhang, Classification strategies for image classification in genetic programming, *Proc. of Image and Vision Computing International Conference*, pp.402-407, 2003.

[10] M. Zhang and W. Smart, Multiclass object classification using genetic programming, *LNCS*, pp.367-376, 2004.

[11] J. K. Kishore, L. M. Patnaik, A. Man and V. K. Agrawal, Application of genetic programming for multicategory pattern classification, *IEEE Transactions on Eolutionary Computation*, 2000.

[12] D. P. Muni, N. R. Pa and J. Das, A novel approach to design classifiers using GP, *IEEE Transactions on Evolutionary Computation*, 2004.

[13] M. Zhang and P. Wong, Genetic programming for medical classification: A program simplification approach, *Genetic Programming and Evolvable Machines*, pp.229-255, 2008.

[14] R. Poli, W. B. Langdo and N. F. McPhee, *A Field Guide to Genetic Programming*, Lulu.com, 2008.

[15] S. M. Winkler, M. Affenzelle 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.

[16] A. Topchy and W. F. Punch, Faster genetic programming based on local gradient search of numeric leaf values, *Proc. of the Genetic and Evolutionary Computation Conference*, pp.155-162, 2001.

[17] M. Zhang and W. Smart, Genetic programming with gradient descent search for multiclass object classification, *The 7th European Conference on Genetic Programming, EuroGP*, pp.399-408, 2004.

[18] J. S. Seo et al., Multimodal function optimization based on particle swarm optimization, *IEEE Transactions on Magnetics*, 2006.

[19] V. G. Gudise and G. K. Venayagamoorthy, Comparison of particle swarm optimization and back propagation as training algorithms for neural networks, *Swarm Intelligence Symposium*, 2003.

[20] M. Clerc and J. Kennedy, The particle swarm explosion, stability, and convergence in a multidimensional complex space, *IEEE Transactions on Evolutionary Computation*, pp.58-73, 2002.

[21] Y. Shi and R. C. Eberhart, A modified particle swarm optimizer, *IEEE Congress on Evolutionary Computation*, pp.69-73, 1998.

[22] Y. Shi and R. C. Eberhart, Particle swarm optimization with fuzzy adaptive inertia weight, *Proc. of the Workshop on Particle Swarm Optimization*, Indianapolis, 2001.

[23] R. Poli, J. Kenned and T. Blackwell, Particle swarm optimizati: An overview, *Swarm Intelligence*, pp.33-57, 2007.

[24] J. Kennedy, Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance, *Proc. of Congress on Evolutionary Computation*, pp.1931-1938, 1999.

[25] P. N. Suganthan, Particle swarm optimizer with neighborhood operator, *Proc. of Congress Evolutionary Computation*, Washington, pp.1958-1962, 1999.

[26] K. E. Parsopoulos and M. N. Vrahatis, UPSO: A unified particle swarm optimization scheme, *Lecture Series on Computational Sciences*, pp.868-873, 2004.

[27] R. Mendes, J. Kennedy and J. Neves, The fully informed particle swarm: Simpler, maybe better, *IEEE Transactions on Evolutionary Computation*, pp.204-210, 2004.
[28] J. Riget and J. S. Vesterstrom, A diversity-guided particle swarm optimizer – The ARPSO, *Technical Report 2002-0*, University of Aarhus, 2002.
[29] A. Silva, A. Neve and E. Costa, Chasing the swarm: A predator pray approach to function optimization, *International Conference on Soft Computing*, 2002.
[30] D. Parrot and X. Li, Locating and tracking multiple dynamic optima by a particle swarm model using speciation, *IEEE Transactions on Evolutionary Computation*, 2006.
[31] T. M. Blackwell and P. J. Bentley, Dynamic search with charged swarms, *Proc. of the Genetic and Evolutionary Computation Conference*, 2002.
[32] M. Rashid and A. R. Baig, PSOGP: A genetic programming based adaptable evolutionary hybrid particle swarm optimization, *International Journal of Innovative Computing, Information and Control*, vol.6, no.1, pp.287-293, 2010.
[33] H. Jabeen and A. R. Baig, DepthLimited crossover in genetic programming for classifier evolution, *Computers in Human Behaviour*, 2010.