BILINGUAL PUBLISHING CO.
Pioneer of Global Academics Since 1984

Artificial Intelligence Advances
https://ojs.bilpublishing.com/index.php/aia

ARTICLE

GFLIB: an Open Source Library for Genetic Folding Solving Optimization Problems

Mohammad A. Mezher*

Dept. of Computer Science, Fahd Bin Sultan University, Tabuk, KSA

ARTICLE INFO

Article history
Received: 8 March 2019
Accepted: 16 April 2019
Published Online: 30 April 2019

Keywords:
GF toolbox
GF Algorithm
Evolutionary algorithms
Classification
Regression
Optimization
LIBSVM

ABSTRACT

This paper aims at presenting GFLIB, a Genetic Folding MATLAB toolbox for supervised learning problems. In essence, the goal of GFLIB is to build a concise model of supervised learning, and a free open source MATLAB toolbox for performing classification and regression. The GFLIB is specifically designed for most of the traditionally used features, to evolve in applications of mathematical models. The toolbox suits all kinds of users; from the users who implemented GFLIB as "black box", to advanced researchers who want to generate and test new functionalities and parameters of GF algorithm. The toolbox and its documentation are freely available for download at: https://github.com/mohabedalgani/gflib.git

1. Introduction

All evolutionary algorithms[1] are biologically stimulated, by using the "survival fittest" concept found with the aid of Darwinian evolution. GF algorithm is one of the EA relative member which is used to resolve complicated problems through randomly producing populations of computer programs. Every computer program (Chromosome) undergoes a number of natural adjustments called crossover and mutation, to create a brand-new population. These operators then could be iterated to generate the fittest chromosome which are evaluated using one of the performance measurements. GF algorithm is a member of the Evolutionary algorithm's family, but it uses a simple floating-point system for genes, in formulating the GF chromosomes.

Certainly, there are quite a number of open source evolutionary algorithms toolboxes used for MATLAB[2, 3], but none specific for genetic folding algorithm. GFLIB looks forward to providing such a free open source toolbox that can be used and developed by others. Accordingly, the GFLIB toolbox was designed from scratch and adopted to

**Corresponding Author:*
Mohammad A. Mezher,
Dept. of Computer Science, Fahd Bin Sultan University, Tabuk, KSA;
Email: mmezher@fbsu.edu.sa

 DOI: https://doi.org/10.30564/aia.v1i1.608

ensure code reusability and clarity. The end result was to deal with a wide variety of machine learning usage problems. The need for a fast and easy way is to try different dataset on a distinct range of parameters. The GFLIB examined on different MATLAB versions and computer systems, namely version (R2017b) for Windows and (R2017a) for Mac.

This standalone toolbox will offer alternatives for users/researchers to help them decide on both training and testing data set with a number of k-folding cross-validation, mathematics operators, crossover and mutation operators, crossover and mutation rates, kernel types, and various number of GF parameters. Furthermore, this toolbox will offer an output option to prevent results in different formats and figures such as; roc curve, structural complexity, fitness values, and mean square errors.

In other words, the aim of building a standalone supervised learning toolbox is to spread GF algorithm in all data set fall within the classification and regression problems.

2. GFLIB Structure

2.1 Previous Version of GF Toolbox Structure

The old version of GFLIB was first introduced having the toolbox rely completely[4] on GP Toolbox[2]. The toolbox contained GF algorithms for supervised classification and regression problems, but it was aligning the structure designated for the GP. At that time, the GF toolbox was lacking from holding unique encoding and decoding mechanisms fully functioning as intended for being integrated with GP Toolbox. The development of GF toolbox, therefore, was oriented to the optimization and integration of the existing GP toolbox. The implementation of GF toolbox was done using MATLAB and the GP package. The idea was to encode and decode using the GF tree wherein, the GFLIB was built using GF mechanism shown in [5].

2.2 Current Version of GFLIB Toolbox Structure

Although the main GF structure was demonstrated in detail in [4, 5], where the paper will be mainly designed to be highlighted on the structure of GFLIB toolbox only. GFLIB is a research MATLAB project that is essentially intended to offer the user with a complete toolbox without the need to know how GF algorithm works on a specific dataset. The recent developed GFLIB toolbox additionally, will grant researchers an entire control in comparison with different well-known evolutionary algorithms. The variety of options which GFLIB presents can be used as a very important tool for researchers, students, and experts who are interested in testing their personal dataset.

2.2.1 Data Structures

GFLIB provides an easy way to add a dataset in a text format. The text files may be found in both UCI dataset [6] and LIBSVM dataset [7]. GFLIB is mainly supporting the .txt data type which is found in the same style of UCI dataset only.

The main data structures in the GFLIB Toolbox are genotype and phenotypes which represents the GF chromosomes. The chromosomes present in GF are considered to be the main structure in the algorithm. The GF chromosome consists three-parts: an index number of the gene in a chromosome which represents the father, and the two points inside the gene which represents the children.

Then the GF chromosome structure encodes an entire population in a single float-number of formats ls.rs, whereas lc is the left child number and rc is the right child number. Phenotypes are stored in a structure of a determined number of populations. The ith population pop (i) consists of chromstr and chromnum, chromstr is formulated for the operator name and chromnum is formulated for the GF encoding number and both represent the lc and the rc. The root operator and GF number must be scalar. In all of these GF structures, each GF number corresponds to a particular gene either for the right child chromosome, or the left child chromosome respectively.

In general, the main purpose of the encoding and decoding process of GF chromosome is to have an arithmetic understanding. GF encodes any arithmetic operation by dividing it to left and right sides. Each side is divided into other valid genes to formulate a GF chromosome. The encoding process depends on the number of operands the arithmetic operations used. At first, two-operands operators' term is (e.g. the minus operator) placed at the first gene, referring to other operators repeatedly to end up with terminals. However, the operator types called by a father gene are; two children (two operands), one child (one operand) and no child (terminal).

In the meantime, to decode a chromosome, take the first gene which has two divisions (children) with respective operands; ls child and rs child. Repeatedly, for each father, a number of children to be called every time until a kernel function is represented. The decoding/encoding process [4, 5, 8, 9] executes the folding father operator (e.g. plus) over the ls child (minus) and the rs child (multiply). The folding mechanisms develops a new algorithm known as Genetic Folding algorithm.

The three datasets used here for comparative analysis includes; Iris dataset (multi-classification problem), a Heart dataset (binary classification problem), and Housing dataset (regression problem). The Iris dataset is a

DOI: https://doi.org/10.30564/aia.v1i1.608

dataset made by biologist Ronald Fisher, used in 1936 as an example of linear discriminant analysis. There are 50 samples from every of 3 species of Iris (Iris setosa, Iris virginica and Iris versicolor). Four features were measured from every sample: the length and the width of the sepals and petals, in centimetres. [10]

The second dataset is the Heart dataset (the part obtained from Cleveland Clinic Foundation), using a subset of 14 attributes. The purpose is to detect heart diseases in a patient. Its integer value goes from 0 (no presence) to 4. [6]

The last dataset for testing the regression problems is "Housing dataset". The Housing dataset has a median value of the house price along with 13 other parameters that could potentially be related to housing prices. The aim of the dataset is to predict a linear regression model by estimating the median price of owner-occupied homes in Boston.

2.2.2 GFLIB Toolbox Structure

The toolbox provides algorithms like SVC, SVR, and Genetic Folding Algorithm. It provides easy to use MATLAB files, which takes in input basic parameters for each algorithm based on the selected file. For example, in regression problems, there is a file (regress.m) to enter kernel type, number of k-fold, number of population and the maximum number of generations to be considered. The list of parameters users can input and uniformed for classification and regression problems are shown in below Table 1:

Table 1. List of Parameters in GFLIB

Name	Definition	Values
Mutprob	mutation probability	A float value
Crossprob	crossover probability	A float value
Maxgen	max generation	An integer value
Popsize	population size	An integer value
Type	problem type	multi,binary,regress
Data	Dataset	*.txt
Kernel	Kernel Type	rbf,linear,polynomial,gf
Crossval	Crossvalidation	An integer value
Oplist	operators and operands	'Plus_s','Minus_s', 'Plus_v','Minus_v','Sine', 'Cosine','Tanh','Log', 'x','y'
Oplimit	length of chromosome	An integer value

The main directory of GFLIB contains a set of main purposes of GFLIB .m files, described in details in this section, and the two following subdirectories;

• **@data** which contains the folder of **@binary** for dataset, **@multi** for multiclassification dataset and **@regress** for regression dataset. The folder to manipulate or add more dataset to these subdirectories.

• **@libsvm**, whose functions, discussed in [7] and being integrated into the toolbox to play the SVM role.

• **Binary, multi** and **regress** files, which forms the basic use of each problem type respectively.

The list of figures shown in Table 2 is designed and integrated into the GFLIB toolbox for the sake of comparison with other algorithms and toolboxes.

Table 2. List of GFLIB Figures Shown in the Toolbox

Name	Type
Population Diversity	Fitness distribution vs. Generation
Accuracy	Accuracy value vs. Generation
Structure Complexity	Tree Depth/size vs. Generation
Tree Structure	GP tree structure
GF Chromosome	GF Chromosome structure

In the developed GFLIB toolbox, the focus was on applying supervised learning to GFLIB toolbox for a real-world problem shown in Table III using LIBSVM as described in Figure 1.

However, the choice on which particular dataset type to be used will be determined by the user referee to it in the path, the GF algorithm will run accordingly. Also, once the user decides on the GF parameters to run with, the right GF algorithm (classifier or regression) will run consequentially.

The GA Toolbox was built using GF structs (chromosomes) for the purpose of implementing the core of GF encoding and decoding mechanisms. Here, the major functions of the GFLIB Toolbox are outlined:

(1) Population representation and initialisation: *genpop, initpop*

The GFLIB Toolbox supports floating-point chromosome representation. The floating-point was initialized by the Toolbox function, to create a floating-point GF chromosome, *initpop*. A *genpop* is provided to build a vector describing the populations and figures statistics.

(2) Fitness assignment: *calcfitnes, kernel, kernelvalue*

The fitness function transforms the raw objective function equations found, using GF algorithm into non-negative values. However, *kernelvalue* which will be repeatedly used for all individuals in the population, *kernel*. The Toolbox supports both *libsvm* [7] package and the *fitrsvm* [11] function in MATLAB. Using both, GFLIB could successfully generate models that are capable of fitting the abalone data set. The result of the libsvm (using the svmtrain function) was used along with svmpredict, to successfully predict the different input parameters. The GF algorithm included eight arithmetic operators in the toolbox. The

 DOI: https://doi.org/10.30564/aia.v1i1.608

arithmetic operators shown in Table 1 are either one operand operator (sine, cosine, tanh, and log) or two operands operator (plus, minus).

(3) Genetic Folding operators: *crossover, mutation*

The GFLIB supports two types of operators by dividing the population size into two-equal sizes. Each half-size will undergo one type of operator. The GFLIB operators are one-point crossover, two-point *crossover*, and swap *mutation* operators.

(4) Selection operators: *selection*

This function selects a given number of individuals from the current population according to their fitness and returns a row structs to their indices. Currently, roulette wheel selection method was conducted for GFLIB toolbox. The selection methods particularly, are required to balance between the quality of solutions and genetic diversity.

(5) Performance figures: *genpop*

The list of figures included to demonstrate the performance of the GF algorithm is; the ROC curve (only for binary), expression tree, fitness values, population diversity, accuracy verses complexity, and structure complexity. The GFLIB also includes well-known kernel functions in order to differentiate comparisons easily. The file also prints the best GF chromosome in two different formats; genes numbers and operator string.

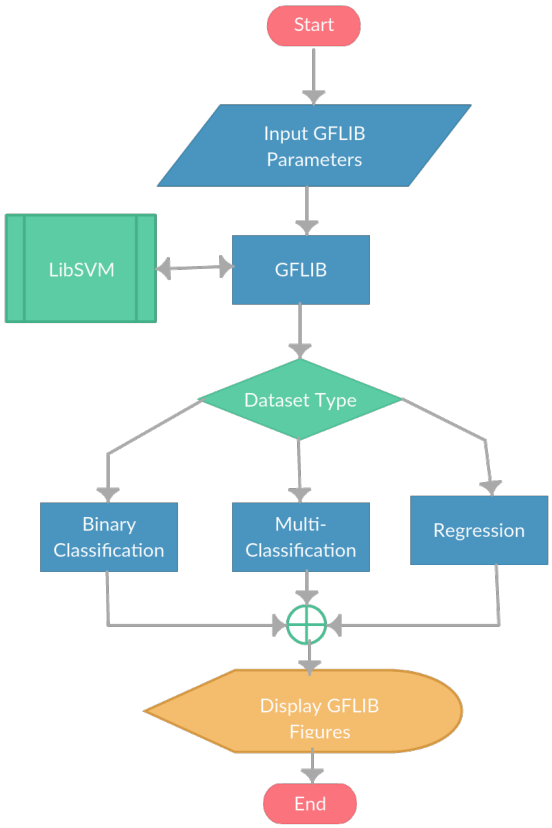


Fig 1. GFLIB life Cycle

3. GF Algorithm Using Generative Models

Genetic Folding (GF) [4, 5, 8, 9] is a novel algorithm stimulated by means of folding mechanism, inspired by the RNA sequence. GF can represent an NP problem by a simple array of floating number instead of using a complex tree structure. First, GF generates an initial population compound of basic mathematics operations randomly. Then, valid chromosomes (expression) can be evaluated. GF assigned a fitness value for every chromosome depending on the fitness function being developed. The chromosome is then selected by the roulette wheel. After which the fittest chromosome will be subjected to the genetic operators in order to generate a new population in an independent way. In every population, the chromosomes are also subjected to a filter to test the validity of the chromosome. The genetic operators used to generate a new population for the next generation. The entire procedure is repeated until the optimum chromosome (kernel) is achieved.

4. Experiments on LIBGF

This paper first shows GFLIB methods work on binary and multi-classification problems; then carries out a regression problem using GFLIB methods. Three datasets are chosen as testing data for the two types of experiments. Part of their properties is included in Table III and Table VI for classification and regression respectively. Amongst them, the same parameters from k-folding to operator list are for experiments conducted. Other well-known kernels are included for the sake of comparison with GFLIB. However, the list of datasets was used in both binary, multi-classification, and regression problems brought from UCI dataset[6].

4.1 LIBGF for Classification Problems

The classification dataset included in GFLIB shown in Table 3 includes the respective details.

Table 3. Classification Datasets Used in the GFLIB

Name	Type	Size
Credit approval	Binary	690*15
Statlog German Credit	Binary	1000*20
Heart Scale	Binary	270*13
Ionosphere	Binary	351*34
Sonar Scale	Binary	208*60
Spam	Binary	4601*57
Iris Scale	Multi	150*4
Zoo	Multi	101*18

The list of parameters' value used in the experiments

DOI: https://doi.org/10.30564/aia.v1i1.608

for both binary and multiclassification problems is shown in table 4.

Table 4. List of Classification Parameter Values

Name	Definition
mutprob	0.1
crossprob	0.5
maxgen	20
popsize	50
type	Binary, multi
data	In table III
kernel	GF, rbf,linear,polynomial
crossval	10-fold
oplist	'Plus_s','Minus_s','Multi_s','Plus_v','Minus_v','x','y'
oplimit	20 %length of chromosome

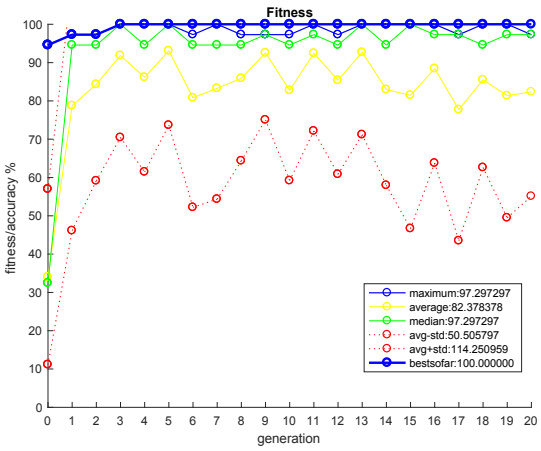


Figure 2. Classification Fitness Values

The best chromosome string found using GFLIB for the iris dataset is:

Plus_s	Sine	Plus_v	X	Y	Sine	Sine	Y	Y	X	X

And the best chromosome GF number formed for the above-mentioned string was:

2.3	4.5	6.7	0.4	0.5	8.9	10.11	0.8	0.9	0.10	0.11

The maximum fitness (Accuracy) found using GFLIB in all generations for the iris dataset was 100.00 %

4.2 LIBGF for Regression Problems

For all figure's types except ROC curve, the experiment was tested on running the algorithm for 20 generations with 10 cross validation. The best performance was the smallest value of the mean square error found of the objective function obtained over all function evaluations. For 50 population conducted at each combination of a half-mutation size, a half-crossover size, 0.1 a mutation rate, and 0.5 a crossover rate. Thus, for each generation, 20 combinations of operators are experimented to form a valid GF chromosome. The GF operators' rates are shown in Table 6.

In Table 5, the list of regression datasets included in GFLIB is shown in and associated with a brief description of the dimensionality.

Table 5. Regression Datasets Used in the GFLIB

Name	Type	Size
Abalone	Regression	4177*8
Housing	Regression	506*13
MPG	Regression	392*6

Table 6 shows the list of parameters and values used to run a regression test on Housing dataset:

Table 6. List of Regression Parameter Values

Name	Definition
mutprob	0.1
crossprob	0.5
maxgen	20
popsize	50
type	regression
data	In table VI
kernel	GF, rbf,linear,polynomial
crossval	10-fold
oplist	'Plus_s','Minus_s','Multi_s','Plus_v','Minus_v','x','y'
oplimit	20 %length of chromosome

The best performance value found was with the MSE value of 0.000121 in all generations as shown in Figure 3. In Figure 4 and Figure 5, the results conducted using the GFLIB to demonstrate the variety of results shown using GFLIB toolbox. The population diversity figure, the figure plots in dots the highest and lowest fitness values found in a population. The structure complexity figure plots the folding depth of the best GF chromosome found in each generation. The size of each folding counted based on the number of calling occurred by the first number formulated in a GF chromosome.

A GF chromosome structure has been well-defined to represent a structural folding of a GF chromosome. Then, the GF chromosome is extracted and arranged as a tree structure of real numbers. The GF encoding part of the toolbox is used to evolve the tree-structure of a program whereas the GF decoding part of the toolbox is applied to determine the string of the structural chromosome. Experimental results have shown the promise of the developed approach.

DOI: https://doi.org/10.30564/aia.v1i1.608

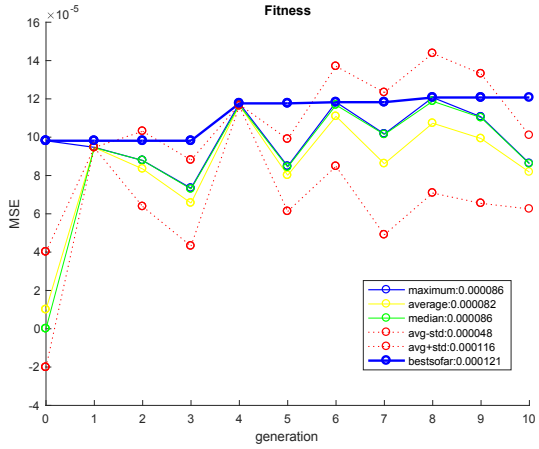
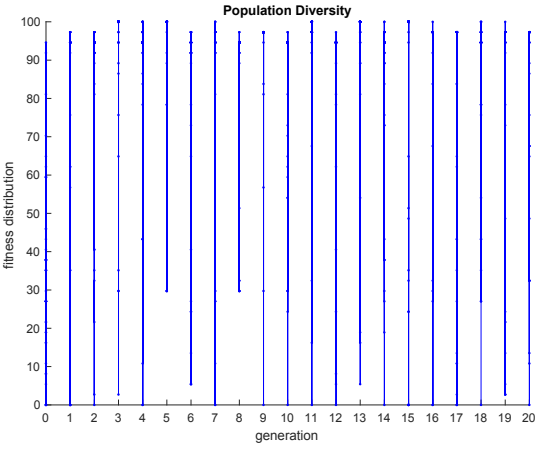
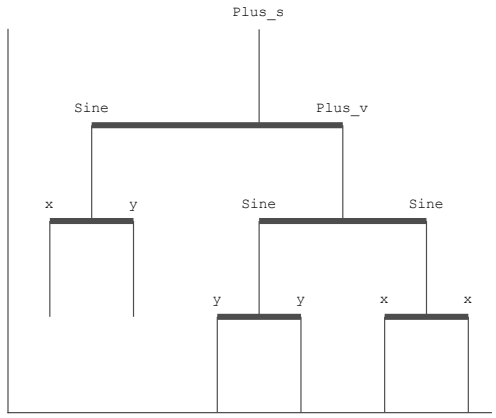


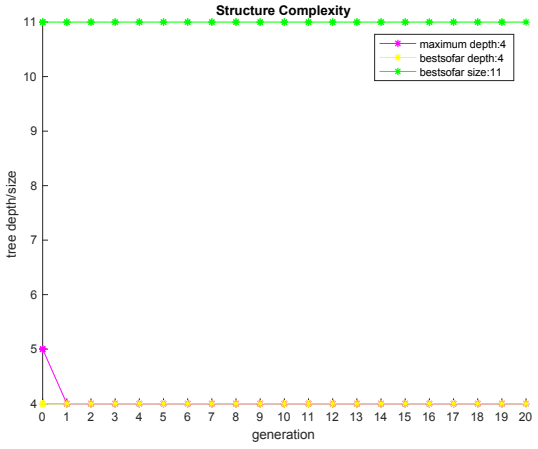
Figure 3. Regression Fitness Value



a. Population Diversity

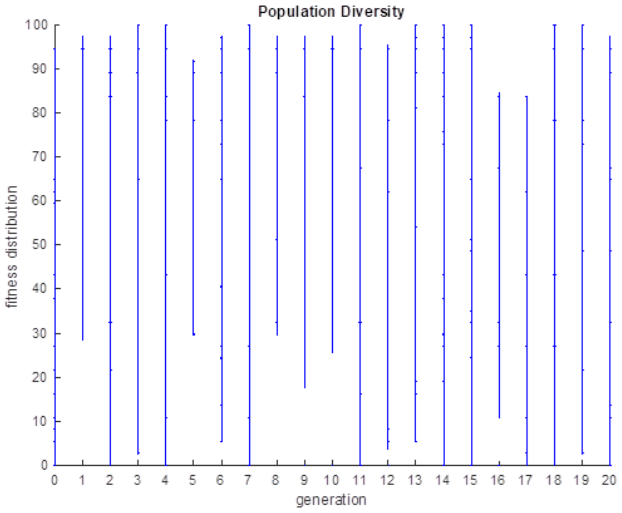


b. GF Tree Structure

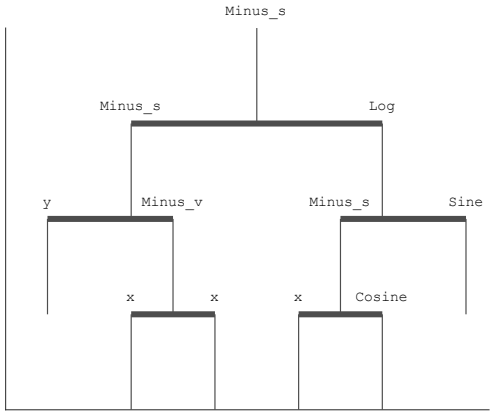


c. GF Structure Complixity

Figure 4. GFLIB Toolbox Ran for Iris Multiclassification Dataset

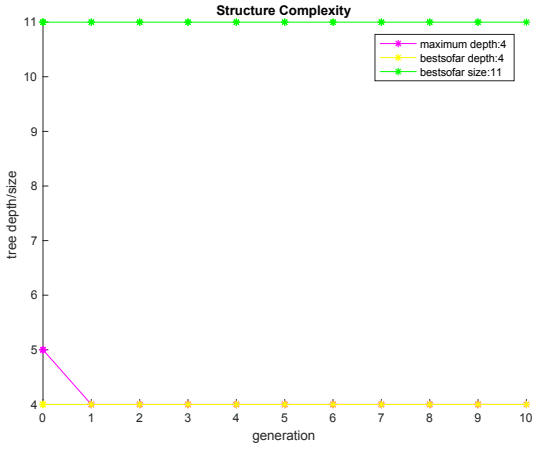


a. Population Diversity



b. GF Tree Structure

DOI: https://doi.org/10.30564/aia.v1i1.608



c. Structure Complixity

Figure 5. GFLIB Toolbox Ran for Housing Regression Dataset

The best GF string found using GFLIB for the iris dataset is:

Minus_s	Minus_s	Log	Y	Minus_v	Minus_s	Sine	X	X	X	cosine

And the best GF number formed for the above-mentioned string was:

2.3	4.5	6.7	0.4	8.9	10.11	0.7	0.8	0.9	0.10	0.11

5. Conclusion

GFLIB toolbox is presented and built using MATLAB, for users and researchers who are interested in solving real NP problems. The key feature of this toolbox is the structure of the GF chromosome, and the encoding and decoding processes included in the toolbox. In this GFLIB toolbox, eleven well-known UCI datasets are studied and implemented with their relative performance analysis; ROC curve, fitness values, structural analysis, tree structure, and population diversity. These datasets can be categorised into two categories: classification and regression. All figures are comparable with another set of three well-known kernel functions. GFLIB toolbox of any category allows users to select their parameter choice. Balanced parameters of GF chromosome must be considered, to maintain the genetic diversity within the population of candidate solutions throughout generations. But, on the other hand, the MATLAB GFLIB files tends to facilitate the development time of the toolbox.

In this paper, the GFLIB is being compared with three well-known kernels. In future researches, I intend to compare GFLIB with a GA and GP alone as well. I also intend to compare the toolbox with other kinds of hybrid methods, such as the hybrid decision tree/instance.

References

[1] Seyedali Mirjalili. Evolutionary Algorithms and Neural Networks Theory and Applications. Springer international Publishing; June 2018.

[2] Sara Silva and Jonas Almeida, “Gplab-a genetic programming toolbox for matlab,” In Proc. of the Nordic MATLAB Conference, pp. 273--278, 2005.

[3] A.J. Chipperfield and P.J. Fleming, “The MATLAB genetic algorithm toolbox”, IEE Colloquium on Applied Control Techniques Using MATLAB, UK, 1995

[4] Mezher, Mohammad and Abbod, Maysam. (2010). Genetic Folding: A New Class of Evolutionary Algorithms. 279-284.

[5] Mohd Mezher, Maysam Abbod. Genetic Folding: An Algorithm for Solving Multiclass SVM Problems. Applied Soft Computing, Elsiver Journal. 41(2):464-472. 2014.

[6] C L Blake, C J Merz. UCI repository of machine learning databases University of California, Irvine, Department of Information and Computer Sciences. 1998.

[7] Chang, Chih-Chung and Lin, Chih-Jen. LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology. 2(3): 1-27. 20011.

[8] Mohd Mezher, Maysam Abbod. Genetic Folding: A New Class of Evolutionary Algorithms. October 2010.

[9]Mohd Mezher, Maysam Abbod. A New Genetic Folding Algorithm for Regression Problems. Proceedings - 2012 14th International Conference on Modelling and Simulation, UKSim. 46-51. 2012.

[10] R. A. Fisher (1936). “The use of multiple measurements in taxonomic problems”. Annals of Eugenics. 7 (2): 179–188.

[11] Statistics and Machine Learning Toolbox Users guide. 2018b,the MathWorks, Inc., Natick, Massachusetts, United States.

 DOI: https://doi.org/10.30564/aia.v1i1.608