Assiut Univ. J. of Mathematics and Computer Science Printed ISSN 1687-4897
49(1), pp.1-17 (2020)

MEMETIC PROGRAMMING WITH THE ATOMIC REPRESENTATION FOR EXTRACTING LOGICAL CLASSIFICATION RULES

Eman Baky, Emad Mabrouk1, I.E. Elsemman

Received: 22/9/2019 **Accepted:** 2/12/2019 **Available Online:** 1/12/2020

Classification is one of the most popular techniques of data mining. This paper presents an evolutionary approach for designing classifiers for two-class classification problems using an enhanced version of the genetic programming (GP) algorithm, called the Memetic Programming (MP) algorithm. MP can discover relationships between observed data and express them logically. MP aims to obtain a classifier with the largest area under the ROC curve, which has been proved a better performance than traditionally metrics. The proposed approach is being demonstrated by experimenting on some UCI Machine Learning data sets. Results obtained in these experiments reflect the efficiency of the proposed algorithm.

Keywords: Classification, Evolutionary Algorithm, Local Search Procedure, Memetic Programming, ROC Curves.

1. INTRODUCTION

The massive growth of data in the real-life applications has driven to development of data mining techniques. Data mining can be defined as the process of discovering knowledge and information from large amounts of data stored in databases [1]. Therefore, researchers considered the data mining as a core step in the process of knowledge discovery from databases [2]. Data mining used many techniques to extract patterns from information, these techniques can be classified into two majors; prediction techniques and description techniques [3]:

Prediction techniques: Use some variables to predict unknown or future values of other variables, such as classification, regression and deviation detection.

Description techniques: Find human-interpretable patterns that de- scribe the data, such as association rules, clustering and outlier analysis.

Classification is a data mining technique used to predict group member-ships for data instances. Specifically, data classification can be defined as allocating class labels to given data instances through two steps [4]:

Building the classifier: In this step, the proposed algorithm finds the re-lationship between values of predictors and values of the target through the training data in which the class assignments are known. We can call this step as the learning step or the learning phase.

Using the classifier: In this step, the algorithm uses a set of testing data to estimate the accuracy of the resulting classification rules. Con- sequently, the classification rules can be applied for the new data if the accuracy is considered acceptable.The genetic classifier is looking for the rule relies on Darwin’s principle of natural selection and operations that mimic naturally occurring genetic operations, such as sexual recombination (crossover) and mutation, see [5, 6, 7]. In this paper, the memetic programming (MP) algorithm will be used to generate rules for a set of classification problems [8, 9].

The rest of the paper is organized as follows: In the next section, we introduce more details about the MP algorithm. The main model for the classification problems are presented in Section 3. In Section 4, we report numerical results for a set of benchmark classification problems. Finally, conclusions make up Section 5.

2. MEMETIC PROGRAMMING

The memetic programming algorithm hybridizes the genetic programming method [10] with a set of local search procedures over a tree space to improve good programs with the highest fitness values. MP inherits the basic idea of memetic algorithms [11, 12, 13, 14], however, MP deals with computer programs. These computer programs are expressed as sparse trees, where internal nodes are called functions and leaf nodes are called terminals, see Figure 1. The user specifies the sets of terminals and functions based on the problem at hand.

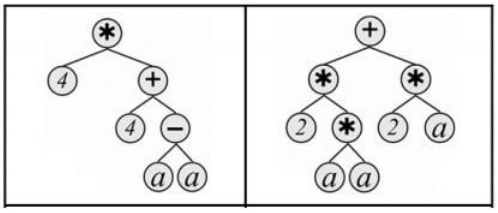

Figure 1: Examples of MP representation.

The main loop of the MP algorithm can be divided into two phases, the diversification phase and the intensification phase. The diversification phase follows the GP algorithm using a suitable selection strategy along with the mutation and crossover operations to guarantee the diversity in the current population. Figure 2 explains an example of applying crossover and mutation operators for some trees. On the other hand, the intensification phase uses a set of local search procedures to intensify promising programs resulting from the diversification phase.

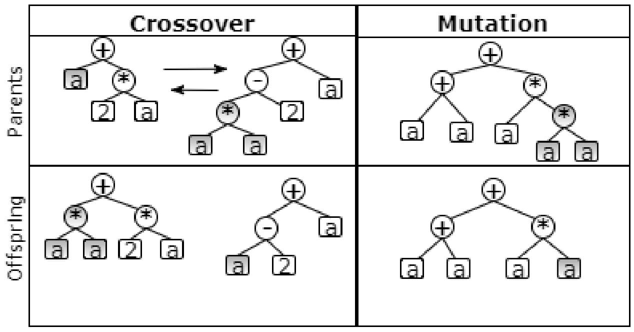

Figure 2: Generating a new offspring using mutation and crossover operators

2.1 Local searches over the tree space

The main objective of the local search operators is to generate new trees in a neighborhood of the selected tree. This subsection discusses two types of local searches; static structure search and dynamic structure search, see Mabrouk et al [8, 9]. The static structure search explores the neighborhood of a tree and modifies some nodes without changing the structure of the original tree, where the shaking operator is employed to perform this job during the static structure search. On the other side, dynamic structure search changes the structure of the tree by extending its terminal nodes or cutting its subtrees, where grafting and pruning operators are used to fulfill the job in the dynamic structure search.

Shaking search is a condensation search procedure that generates a new tree X˜from a tree X by altering some terminals or/and some functions chosen randomly, without changing the structure of X. Specifically, the terminal node is altered by another terminal and the function node is altered by another function node with the same number of arguments.

Grafting search is a diverse local search procedure that generates a new tree X˜ from a tree X by extending some of its terminals to be branches ،where the terminals are chosen randomly and the new branches are generated randomly with depth ζ . Pruning search is another diverse local search procedure. In converse with the grafting search, the pruning search generates a new tree X˜ from a tree X by cutting some of its branches of depth ζ ≥ 1 and replacing them by new terminals, where the branches and terminals are chosen randomly. Figure 3 shows three examples of generating set of new trees by applying shaking, grafting and pruning procedures.

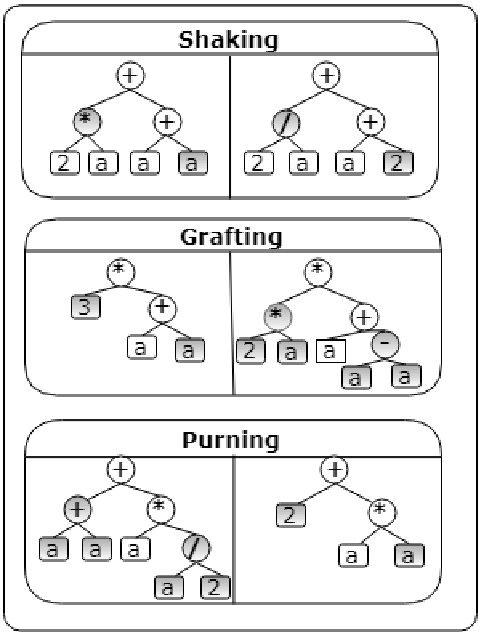

Figure 3: Generating new trees using shaking, grafting and pruning procedures

2.2 Local search programming algorithm

A local search algorithm over a tree space, called the local search program- ming (LSP) algorithm is proposed to find the best program in the neighbor- hood of a given tree. The LSP algorithm employs the local search procedures, in Subsection 2.1, to generate new solutions in the neighborhood of an elected solution and iterates the process as long as it improves the solution under consideration. The LSP process will be terminated if no better solutions can be found in the neighborhood of the current one. Figure 4 shows the flowchart of the LSP algorithm.

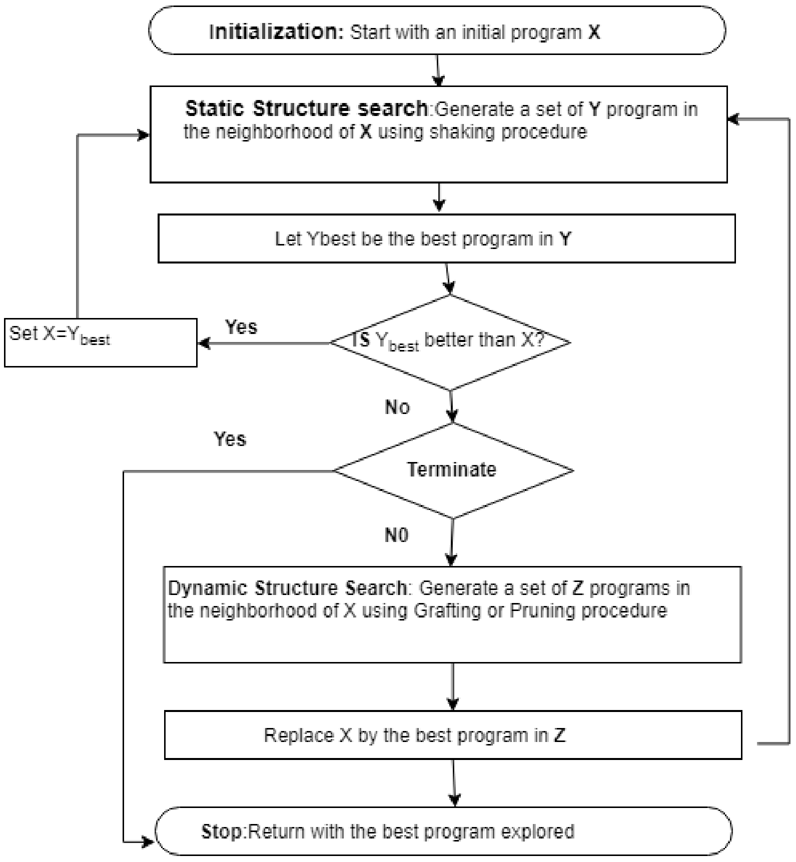
Initialization: Start with an initial program X
Static Structure search:Generate a set of Y program in the neighborhood of X using shaking procedure
Let Ybest be the best program in Y
IS Ybest better than X?
Yes
Set X=Ybest
No
Yes
Terminate
NO
Dynamic Structure Search: Generate a set of Z programs in the neighborhood of X using Grafting or Pruning procedure
Replace X by the best program in Z
Stop:Return with the best program explored

Figure 4: The flowchart of LSP.

3. THE PROPOSAL MODEL

In this section we introduce the proposed model for solving two-class classification problems. Mainly, the model is consisting of three stages; data preprocessing, data cross-validation and generating the classifier, Figure 5.

Subsection 3.1 introduces some necessary preparations for data under consideration. Using the new dataset, MP will be applied to generate a classifier with highest fitness value. Then, the accuracy of this classifier will be measured using the testing dataset as discussed in Subsection 3.2.

3.1 Data preprocessing

All classification algorithms make some special preparations on the given datasets before using it in the search process. In this subsection we introduce some of these data preprocessing techniques that will be used through the experimental results in this paper.

Data cleaning: In the real-life applications we sometimes must deal with incomplete, noisy, and inconsistent data. Therefore, additional treatments must be apply before using the dataset by the data mining techniques. Data cleaning aims to clean up the dataset under con- sideration by filling in missing values, smoothing out noisy data and correcting inconsistency in the given data, if any.

Data transformation: Many machine learning models require transforming nominal variables in the given dataset to numeric variables. Indeed, values of a nominal attribute are given as strings and represent different names, i.e., zip codes and eye color. To solve this problem, we transform the set of nominal values to a new set of binary attributes. Therefore, N different nominal values can be represented as N different binary numbers.

Data normalization: Normalization refers to scale all values of numeri- cal attributes under consideration to fall within a predefined min-max range. In this paper, the following equation used to transform the at- tribute x to the new attribute z, where all values of z lie in the interval [0, 1]:

$$x = \frac{\mathrm{x} - \min(\mathrm{x})}{\max(\mathrm{x}) - \min(\mathrm{x})} \tag{1}$$

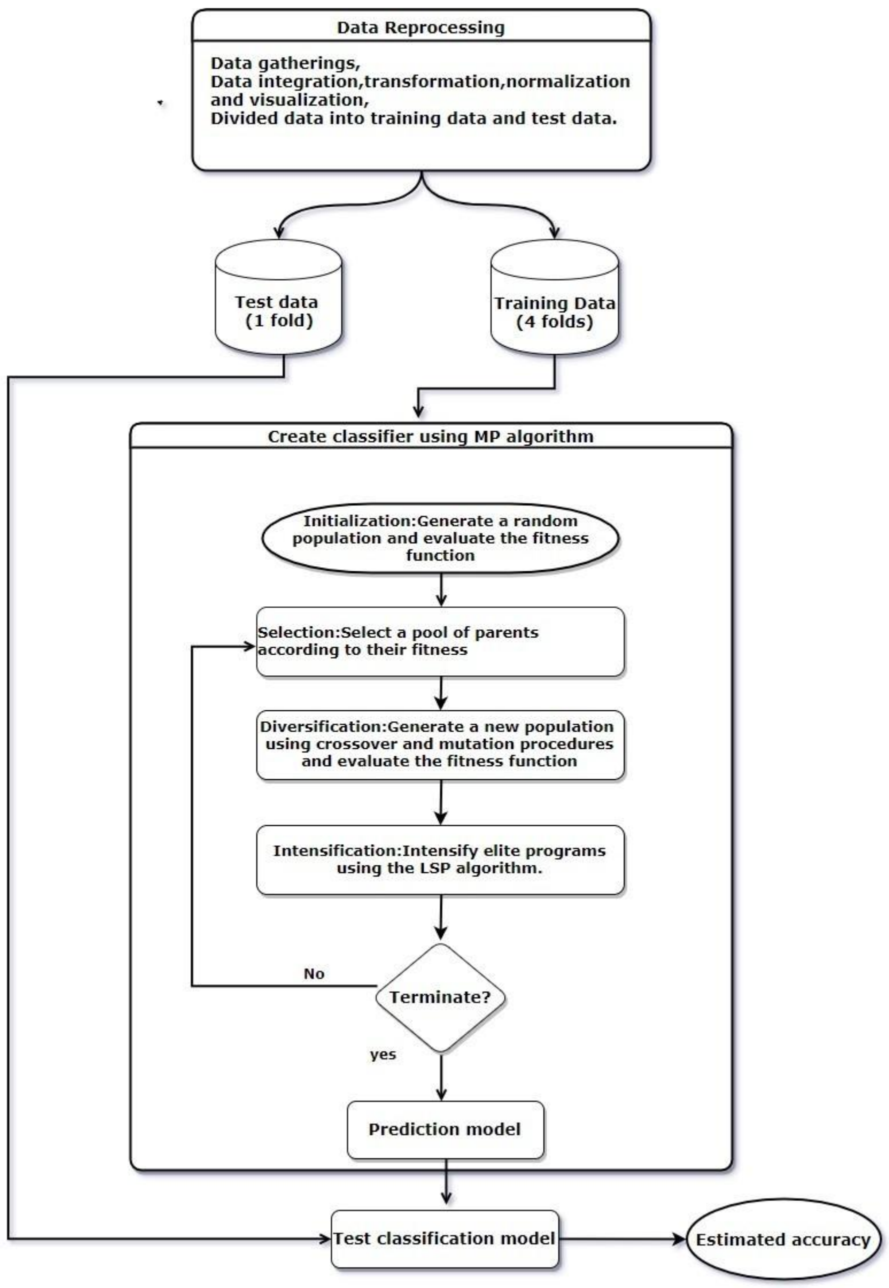
Data Reprocessing
Data gatherings,
Data integration,transformation,normalization and visualization,
Divided data into training data and test data.
Test data (1 fold)
Training Data (4 folds)
Create classifier using MP algorithm
Initialization:Generate a random population and evaluate the fitness function
Selection:Select a pool of parents according to their fitness
Diversification:Generate a new population using crossover and mutation procedures and evaluate the fitness function
Intensification:Intensify elite programs using the LSP algorithm.
No
Terminate?
yes
Prediction model
Test classification model
Estimated accuracy

Figure 5: The flowchart of proposed model

Data cross-validation: Cross-validation is a statistical technique used to assess the ability and stability of machine learning models. Cross- validation technique divides the given dataset into two complementary subsets, the training set and the testing set. The training set will be em- ployed to train the model to generate the required classifier. Then, the testing set can be used to validate the stability of the model. K fold cross-validation is one of the most famous cross-validation techniques. In K fold cross-validation, the entire dataset is randomly split into K folds, with K 1 folds are used as the training dataset, and the remaining fold is retained as the validation or testing dataset. Then the model can generate a classifier using the training dataset and the error will be estimated using the testing dataset. This process is then repeated K times until each of the K folds is used exactly once as the testing dataset. The average of the resulting K recorded errors, called the cross-validation error, will be considered as the performance metric for the model. During the experimental results of this paper, the K−fold cross-validation technique will be used with K = 5.

3.2 Creating a classifier

The MP algorithm will be used to generate the required classifier according to procedures in Section 2. In this subsection, we explains the representation of a solution in MP, and the fitness function used in this implementation.

3.2.1 Solution representation

For each classification problems the set of functions and the set of terminals must be determined before calling the algorithm. The MP algorithm proposed in this paper use the atomic representation, where each terminal node of a tree is an atom contains three arguments, attribute name, relational operator (<,>,=), and an attribute value. It means that, the atom is syntactically a predicate of the form operator(variable, operator, value). Figure 6 show the atomic representation, where the atom returns true if the condition is satisfied and false otherwise.

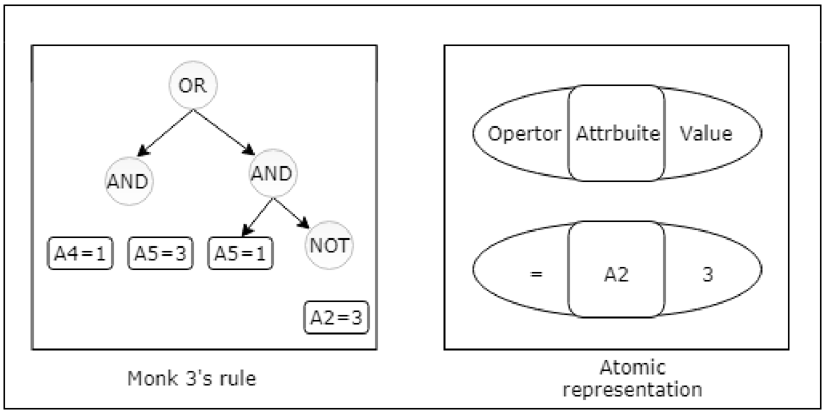

Figure 6: Atomic representation.

3.2.2 Fitness function

The fitness function plays an important role and guides the search process in evolutionary algorithms (EAs). The main objective of defining a fitness function is to evaluate the quality of generated solutions of the proposed algorithm. Many fitness function formulas can be defined for the classification problem, however the most famous one [6, 15][9,10] is as the following:

$$\text{Fitness} = SE * SP, \tag{2}$$

$$SE = TP/(TP + FN) \tag{3}$$

$$SP = TN/(TN + FP), \tag{4}$$

where:

True positive (TP): The number of examples for which the rule returns true and the class label is positive.

False positive(FP): The number of examples for which the rule returns true and the class label is negative.

True negative(TN): The number of examples for which the rule returns false and the class label is negative.

False negative(FN): The number of examples for which the rule returns false and the class label is positive.

4. NUMERICAL EXPERIMENTS

In this section, a set of benchmark problems are considered and tested to estimate the efficiency of the proposed version of MP algorithm.

4.1 Dataset evaluations

To validate our algorithm, 10 datasets of the two-class classification problem are used from the UCI datasets [16], see Table 1. Three datasets, the Monk's problems, are consisting of nominal attributes, and the remaining datasets are consisting of nominal and continuous attributes. Some data preprocessing are applied for these datasets before running the MP algorithm. All missing values are replaced with statistical values, the average values for continuous attributes and the mode values for binary and nominal attributes. We filled 16 and 25 continuous attributes in the breast cancer and credit datasets, respectively. Additionally, We filled 30, 2480 nominal attributes in credit and mushroom datasets, respectively. The parameter values of the MP algorithm are chosen based on several experiments as in Table 2. These set of parameters of the MP algorithm can be summarized as the following:

- nPop: The population size.
- nGnrs: The maximum number of generations.
- nLs: The number of programs selected to apply local search.

nTrs: The number of trial programs generated in the neighborhood of the selected program.

nFails: The maximum number of non-improvements for each call of the LSP algorithm.

- MaxDepth: The maximum depth of a tree.

The 4-way tournament selection is used as the main selection strategy of the MP algorithm, where the algorithm selects four classifiers randomly and run a tournament among them. The fittest classifier of those selected classifiers is chosen to generate some offspring for the next generation.

Table 1: UCI Data Sets.

Data Set	No. Attributes	No. Instances
Credit Approval	15	690
Breast Cancer Wisconsin	10	699
Statlog (Heart)	13	270
MONK's Problems	7	432
Mushroom	22	8124
Voting Records	16	435
Pima Indians Diabetes	8	768
Tic-Tac-Toe Endgame	9	958

Table 2: The parameter values for MP algorithm

Parameter	Value
nPop	500
nGnrs	500
nLs	2
nTrs	3
nFails	4
MaxDepth	5

4.2 Logical classifier for the Monk's problems

The Monk's problems appear in the artificial robot domain, where robots are described by six different nominal attributes; a1, a2, a3, a4, a5, a6 and the 7th attribute is the class label of each sample [17]. The exact solutions for each Monk's problem is known, therefore these problems can be used to estimate the performance of a classifier precisely. For Monk 1 problem the exact solution is (a1 = a2) or (a5 = 1) , and the exact solution of Monk 2 problem is exactly two of its six attributes have their first value, i.e. two

the exact solutiΣon of Monk 3 problem is { (a5 = 3) and (a4 = 1) or (a5 *f*=

In this experiment, 10 independent runs of the proposed classifier are performed for each one of Monk's problems and best solutions found are shown in Figures 7, 8 and 9. These solutions can be reduced and simplified to get the exact solutions for Monk 1, Monk 2 and Monk 3 problems, respectively. Moreover, training datasets of Monk 2 and Monk 3 contain 5% noise. The accuracy of these solutions are shown in Table 3 along with some results in the literature.

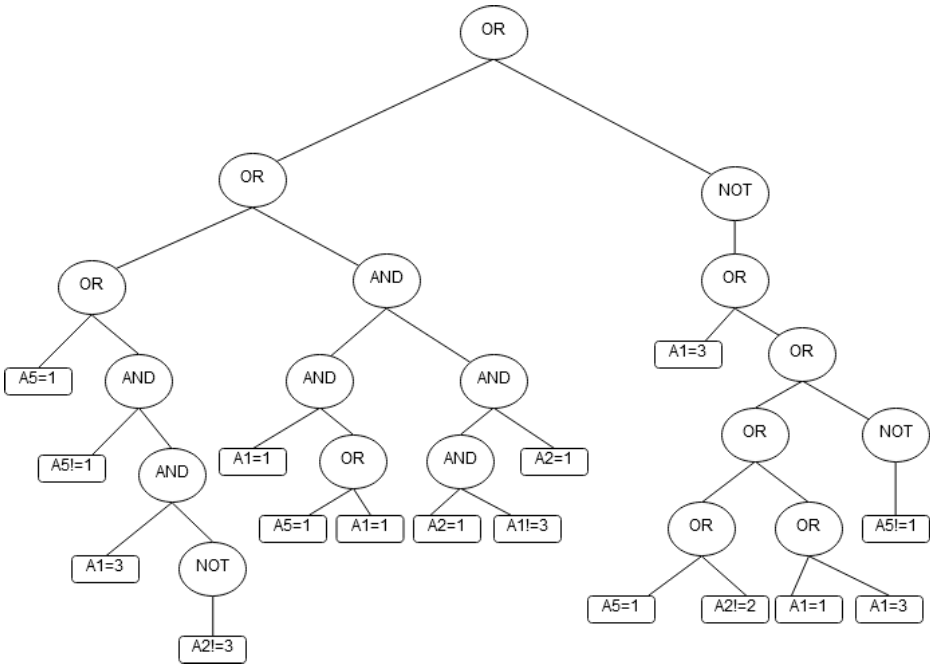

Figure 7: Best solution of Monk 1 problem.

To the best of our knowledge, results of our classifier is the highest classifier extracting logical classification rules for Monk #2. Wong and Leunga applied Grammar-GP and found 65% accuracy [19]. El-Semman and Hassan reported 65%, 65.2% and 71.52% accuracy using C4.5, C4.5 Rules and GEP methods [17]. Marghny reported 100%, 99.40%, 95.90% accuracy using Neu- ral network with genetic algorthim. MPC was applied by Farhat et al.[20] found 91.66%, 81.01%, 88.88% accuracy. However, Pan and Jiao extracted mathematical classification rules found with the GAEC method with 79.28% accuracy [21].

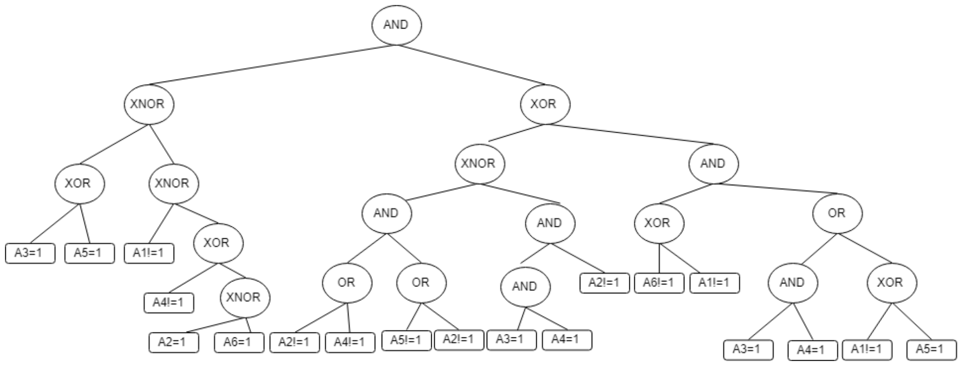

Figure 8: Best solution of Monk 2 problem.

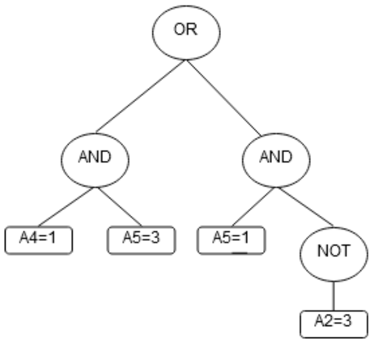

Figure 9: Best solution of Monk 3 problem Table 3: Results and comparisons for Monk's problems.

3.4 Logical classifier for problems with continues attributes

In the previous Subsection, we applied the proposed classifier for Monk's problems with nominal attributes. To validate our classifier on problems with continues attributes, 7 datasets are selected from the UCI website. Table 1 shows the features of these datasets. However, Table 4 shows the accuracy of the best solution found by the proposed classifier for each dataset.

Results and comparisons for Monk's problems

Methods	Monk 1	Monk 2	Monk 3
C4.5	75.70%	65%	97.90%
C4.5Rules	100%	65.20%	96.3%
Grammar-GP	100%	65%	95.4%
GEP	100%	71.52%	97.22%
GAEC	100%	79.28%	100%
MPC	91.66%	81.01%	88.88%
Neural network with genetic algorthim [18]	100%	99.40%	95.90%
Proposed MP	100%	99.07%	100%

Table 4: Results of the MP algorithm for the UCI datasets.

Datasets	Accurac y	S. Deviation
credit	98.55	0.63
breast cancer	100	1.47
heart	100	0.79
monk1	100	5.72
monk2	99.07	4.45
monk3	100	1.86
mushroom	100	0.52
vote	100	0.35
pima	83.12	1.20
tic-tac-toe	96.88	2.80

Figure 10 illustrates the ROC curves for datasets under consideration. These curves reflect the ability of the generated model to distinguish between classes [22]. The ROC curve is plotted based on the true positive rate (TPR) on the Y-axis against the false positive rate (FPR) on the X-axis, where TPR = SE and FPR = 1 SP . From these ROC curves, we can argue that the proposed algorithm can produce excellent and efficient classifiers since the area under the ROC curves near to 1.

5. CONCLUSIONS

In this paper, the Memetic Programming (MP) algorithm has been used for producing mathematical rules for the two-class classification problems.

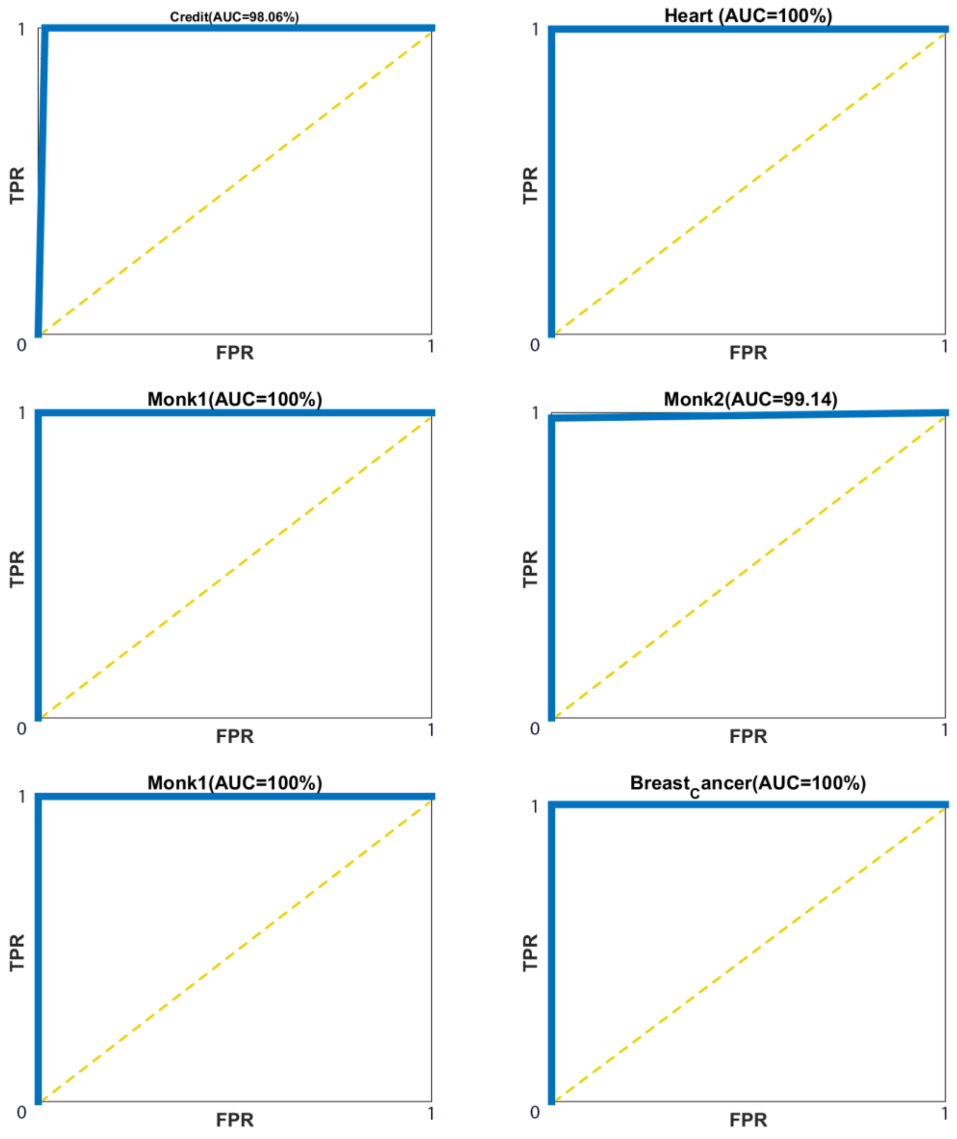

Figure 10: ROC curves for UCI datasets.

The proposed algorithm has been tested to generate new classifiers for a set of benchmark problems from UCI datasets. These datasets have been classified to two types of classification problems, datasets with nominal attributes and dataset with continues attributes. The results of these experiments reflects the efficiency of the MP algorithm compared with other algorithms in the literature at least for the considered benchmark problems.

REFERENCES

1. J. Han, J. Pei, M. Kamber, Data mining: concepts and techniques, Elsevier, 2011.
2. U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy, et al., Advances in knowledge discovery and data mining, Vol. 21, AAAI press Menlo Park, 1996.
3. M. Kantardzic, Data mining: concepts, models, methods, and algo- rithms, John Wiley & Sons, 2011.
4. L. Breiman, Classification and regression trees, Routledge, 2017.
5. P. J. Rauss, J. M. Daida, S. Chaudhary, Classification of spectral im- agery using genetic programming, Ann Arbor 1001 (2000) 48109.
6. C. C. Bojarczuk, H. S. Lopes, A. A. Freitas, Genetic programming for knowledge discovery in chest-pain diagnosis, IEEE Engineering in Medicine and Biology Magazine 19 (4) (2000) 38–44.
7. N. S. Chaudhari, A. Purohit, A. Tiwari, Genetic programming for classi- fication, International Journal of Computer and Electronics Engineering, IJCEE 1 (2009) 69–76.
8. E. Mabrouk, A.-R. Hedar, M. Fukushima, Memetic programming with adaptive local search using tree data structures, in: Proceedings of the 5th international conference on Soft computing as transdisciplinary sci- ence and technology, ACM, 2008, pp. 258–264.
9. E. Mabrouk, A. Hedar, M. Fukushima, Memetic programming algo- rithm with automatically defined functions, Tech. rep., Technical Re- port 2010-015, Department of Applied Mathematics and Physics, Kyoto University, Japan (2010.(
10. J. R. Koza, Genetic programming: on the programming of computers by means of natural selection, Vol. 1, MIT press, 1992.
11. W. E. Hart, N. Krasnogor, J. E. Smith, Recent advances in memetic algorithms, Vol. 166, Springer Science & Business Media, 2004.
12. O. Kramer, Iterated local search with powells method: a memetic al- gorithm for continuous global optimization, Memetic Computing 2 (1) (2010) 69–83.
13. N. Krasnogor, J. Smith, A tutorial for competent memetic algorithms: model, taxonomy, and design issues, IEEE Transactions on Evolutionary Computation 9 (5) (2005) 474–488.
14. P. Moscato, et al., On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms, Caltech concurrent com- putation program, C3P Report 826 (1989) 1989.

15. C. C. Bojarczuk, H. S. Lopes, A. A. Freitas, E. L. Michalkiewicz, A constrained-syntax genetic programming system for discovering classification rules: application to medical data sets, Artificial Intelligence in Medicine 30 (1) (2004) 27–48.
16. A. Asuncion, D. Newman, Uci machine learning repository (2007.(
17. M. Marghny, I. El-Semman, Extracting logical classification rules with gene expression programming: microarray case study, in: Proceedings of the International Conference on Artificial Intelligence and Machine Learning (AIML 05), 2005, pp. 11–16.
18.]M. H. Mohamed, Rules extraction from constructively trained neural networks based on genetic algorithms, Neurocomputing 74 (17) (2011) 3180–3192.
19. M. L. Wong, K. S. Leung, Data mining using grammar based genetic programming and applications, Vol. 3, Springer Science & Business Media, 2006.
20. A. A. Farhat, I. El-Semman, E. Mabrouk, Solving two-class classification problem using memetic programming, Assiut Univ. J. of Mathematics and Computer Science (14.(
21. X. Pan, L. Jiao, A granular agent evolutionary algorithm for classifica- tion, Applied Soft Computing 11 (3) (2011) 3093–3105.
22. W. J. Krzanowski, D. J. Hand, ROC curves for continuous data, Chap- man and Hall/CRC, 2009.