Self-adjusting Associative Rules Generator for Classification : *An Evolutionary Computation Approach*

K. Lavangnananda
School of Information Technology
King Mongkut's University of Technology Thonburi (KMUTT),
126 Pra-cha-u-tid Road, Bangmod, Bangkok 10140, **Thailand**.
E-mail: kitt@sit.kmutt.ac.th

Abstract — **The problem of generating efficient association rules can seen as search problem since many different sets of rules are possible from a given set of instances. As the application of evolutionary computation in searching is well studied, it is possible to utilize evolutionary computation in mining for efficient association rules. In this paper, a program known as Self-adjusting Associative Rules Generator (SARG) is described. SARG is a data mining program which can generate associative rules for classification. It is an improvement of the data mining program called Genetic Programming for Inductive learning (GPIL). Both utilize evolutionary computation in inductive learning. The shortcoming of GPIL lies in the operations crossover and selection. These two operations were inflexible and not able to adjust themselves in order to select suitable methods for the task at hand. SARG introduces new method of crossover known as MaxToMin crossover together with a self-adjusting reproduction. It has been tested on several benchmark data sets available in the public domain. Comparison between GPIL and SARG revealed that SARG achieved better performance and was able to classify these data sets with higher accuracy. The paper also discusses relevant aspects of SARG and suggests directions for future work.**

I. INTRODUCTION

Applications of artificial intelligence in data mining [1] have received much attention in the past few years. A major benefit of data mining includes the ability to cluster or generate patterns from large amount of data when conventional statistical methods are proven ineffective. One of the most popular methods to cluster the data is by means of generating association rules [2]. Inductive learning is one of many techniques in this sub-field where patterns can be learned or induced from a collection of samples [3]. The popular method in evolutionary computation employed in inductive learning has been Genetic Algorithms (GAs). Another well known technique in evolutionary computation, Genetic Programming (GP), have also been introduced to this area recently.

This paper begins with a brief introduction to how evolutionary computation and inductive learning can be applied to mining instances for associative rules. This is followed by a review of related work. Genetic Programming for Inductive learning (GPIL) is first explained, as it was the basis for SARG. Improvement to GPIL in SARG is described in the following section. Data sets used and the results of their classification using SARG are then shown. Finally, the paper is concluded and future work is suggested.

II. ASSOCIATIVE RULES AND INDUCTIVE LEARNING

The knowledge discovery method *association rules* was first introduced in [2] and was commonly known as the *market-basket problem*. Association rules have an application in clustering and, in particular, classification. From this perspective, the problem of generating minimum and most efficient association rules can also be seen as a search problem as many possible set of rules are possible.

Evolutionary computation can be considered as a searching method [4], it is applicable and appropriate where the search space is very large and exhaustive search is not feasible. Inductive learning [5] is a well known machine learning technique where decision trees can be built from a given data set. These trees can be converted into a set of IF-THEN rules, and hence inductive learning may be employed to generate associative rules. Using inductive learning algorithm in classification has an advantage over technique such as neural networks since decision trees or rules are understandable to human where as numerical weight values in neural networks cannot easily be deciphered.

III. Related Work

There have been numerous inductive learning tools, such well known tools include C4.5 [6] and CART [7] where decision trees are produced. Comparative study between decision tree induction and genetic algorithms can be found in [8]. The first two applications of evolutionary computation to inductive learning were Genetic Algorithm Batch concept Learner (GABL) [9] and Genetic Algorithm Batch-Incremental concept Learner (GABIL) [10]. These two algorithms represent rules by chromosomes of binary strings where length of chromosomes were kept constant.

The application of evolutionary computation in the production of association rules is plentiful. Two examples of recent work include [11] which studied constrained-syntax genetic programming for classification rules. Another is in bioinformatics [12] where cooperative genetic algorithms are executed in parallel to introduce diversity to the solution. There have been several works which utilize genetic programming with inductive learning for various applications [13]. Extensive review of related works can be found in [14].

IV. Genetic Programming for Inductive Learning (GPIL)

The program which lies the foundation for this work is the program Genetic Programming for Inductive Learning (GPIL) [15]. GPIL was motivated by the early program known as Genetic Algorithm for Inductive Learning (GAIL) [16], its application is in classification where chromosomes represent classification rules. GAIL had been tested on benchmark data sets such as the IRIS flower classification [17] and the Cleveland heart data [18]. GAIL was furthered improved into SynGAIL [19], which achieved higher accuracy than GAIL, by adopting synergism of different GAIL units. The shortcoming of both GAIL and SynGail are their use of fixed length chromosomes by binary bits. Although they may be overcome, to some degrees, by the use of *don't care* bits, it makes representation of the rules more tedious and increases computation complexity. This shortcoming is more effectively overcome in GPIL by representation of chromosomes by a tree as used in genetic programming. GPIL comprises 3 main components, *data preprocessing, evolutionary computation* and *final rule builder*, as shown in Figure 1.

A. Data preprocessing

The data set must be split into 2 sets, training set and test set. This component preprocesses the training set for the input to the evolutionary computation.

B. Evolutionary Computation

This component is responsible for generating rules. It comprises 2 processes as namely, *Regrouping* and *Genetic Programming* (GP). Referring to Figure 1, the number of Regrouping, GP and Rules sorting sequences are equal to the number of categories in the classification. Each sequence is responsible for generating rules for that

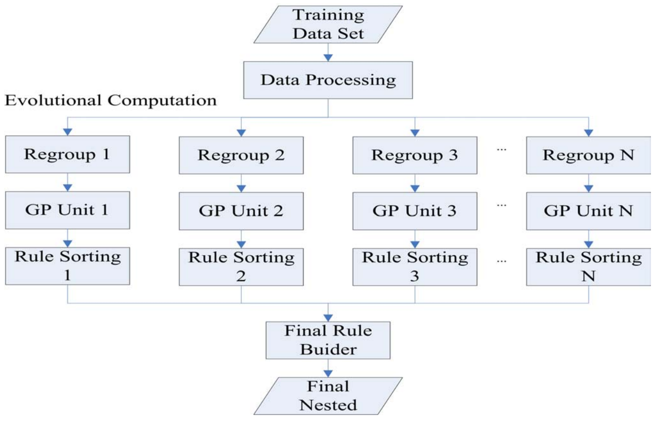


Fig.1. Three main components in GPIL

category. The main process in each GP Unit is described in Figure 2.

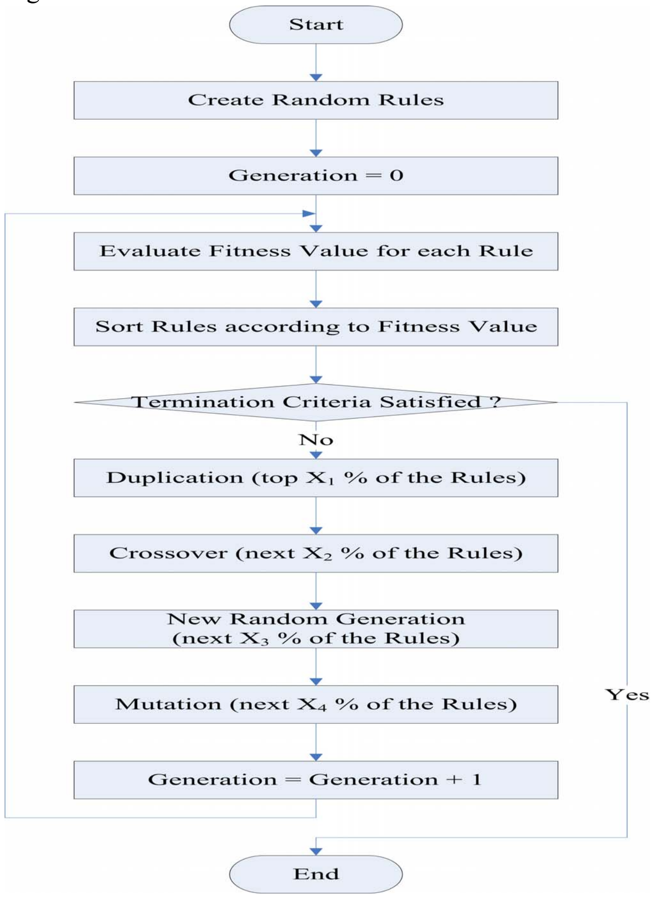

Start
Create Random Rules
Generation = 0
Evaluate Fitness Value for each Rule
Sort Rules according to Fitness Value
Termination Criteria Satisfied ?
No
Duplication (top X₁ % of the Rules)
Crossover (next X₂ % of the Rules)
New Random Generation (next X₃ % of the Rules)
Mutation (next X₄ % of the Rules)
Generation = Generation + 1
Yes
End

Fig. 2. Evolutionary Computation in GPIL

Values of X_1 to X_4 were percentile allowed for each genetic operation (i.e. $X_1 + X_4 + X_3 + X_4 = 100$). These values can be specified by user. The *Final Rule Builder* will be discussed in Section V.

V. Self-adjusting Associative Rule Generator (SARG)

Although GPIL is a considerable improvement to GAIL and SynGAIL, numerous experiments and careful analysis had revealed some drawbacks and limitations. They can be summarized as follows :

Limited crossover method : As size of a tree which represents a chromosome may vary. The position (i.e. node) in a tree, which crossover takes place, is selected randomly. Therefore, the success of reproducing new chromosomes with higher fitness values is left to chance too much.

Rigid selection method : The selection method in GPIL is *rank selection*. While this is sufficient in ordinary cases, it is unproductive in situations, similar to plateau in hill climbing search [20], where progress cannot be made after number of generations. Hence, better rules cannot be generated.

Self-adjusting Associative Rules Generator (SARG) is developed to overcome these drawbacks and limitations. During the inductive learning, SARG has the ability to adjust its evolutionary computation to suit the nature of the data set (i.e. the training set) at hand. This ability is made possible by incorporation of two methods, *MaxToMin Crossover* and *Self-adjusting Evolutionary Computation.*

A. *MaxToMin Crossover*

Adopting different crossover methods in evolutionary computation systems is not uncommon and have been incorporated into systems where binary bits were used to represent populations. Recent example can be found in [21] where individuals are categorized by their fitness value into three categories, suitable crossover rate is then assigned to each category accordingly.

In GPIL, each chromosome (i.e. tree) is evaluated for its fitness value, this value represents the fitness of the whole tree. Hence, fitness is assigned to the composite structure rather than at each component (i.e. node) of the tree. If fitness value is assigned at component level, then it is possible to analyze at the strength and the weakness of each branch in a tree. Therefore, less favourable branches may be pruned off and more favourable ones can be retained rather than accepting or rejecting a whole tree.

In MaxToMin crossover, it is believed that a new tree, which is the result of pruning off the least favourable branch and replacing it with the most favourable branch, ought to have a better chance of being fitter than its predecessors. In SARG, a possible association rule is represented by a tree. A tree comprises interior and terminal nodes, terminal nodes are possible values for attributes and interior nodes are boolean opeartors. The fitness value of a tree is a measure which assesses how accurate it can classify samples in the training set. Fitness value is also determined at each node too. Since each terminal node represent a value of an attribute, fitness value is assigned proportionally to frequency of its appearance in the training set. Similarly, fitness value is assigned to each interior node proportionally to number of samples in which that condition is satisfied. For example, assuming that a training set contains 4 samples; Sample 1: ACFJ, Sample 2: BCGK, Sample 3: ADGK and Sample 4: AENK. Figures 3(a) and 3(b) depict the MaxToMin crossover.

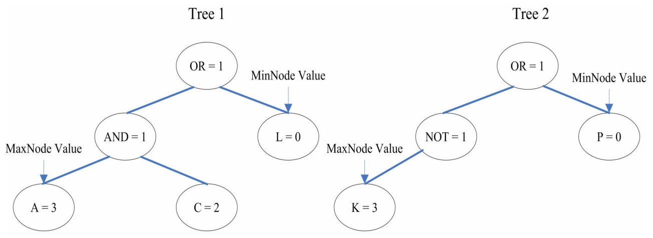


Fig. 3(a). Two arbitrary trees before the MaxToMin crossover

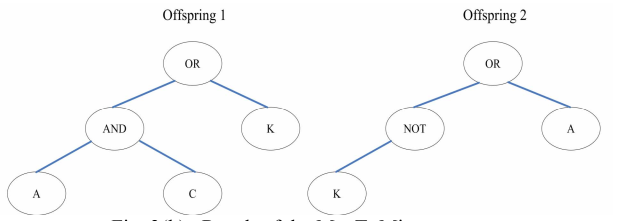


Fig. 3(b). Result of the MaxToMin crossover

B. *Self-adjusting Evolutionary Computation*

An ability to adapt itself to suit a unique task at hand is advantageous to any evolutionary computation system. Several approaches have been suggested in literature [13], [14]. Recent approach includes a system where population is divided into hierarchy of two classes, *elite population* and *plain population* [22]. It was found to be superior than a system with single-type population.

One of the obstacles to progress in evolutionary computation is plateau where improvement cannot be made. In order to rectify this, modification to the current routine or operations is necessary. However, it must be borne in mind that this is under an assumption that an improvement is possible, since improvement may not at all be possible in a problem which is inherently too hard. Genetic programming (GP) units in SARG are self-adjusting. Three different modes are made possible according to the progress in the GP units. These three modes are shown in Table 1. Mode 1 is the original techniques adopted in GPIL. They are selected by default initially. If a GP unit progresses well (i.e. better and better association rules are being generated), Mode 1 is satisfactory and hence no adjustment is required.

TABLE 1.
THREE MODES IN SARG

	Mode 1	***Mode 2***	***Mode 3***
Crossover	Random	Random	MaxToMin
Selection	Rank	Random	Rank

Mode 1 is the original techniques adopted in GPIL. They are selected by default initially. If a GP unit progresses well (i.e. better and better association rules are being generated), Mode 1 is satisfactory and hence no adjustment is required. The progress in a GP unit is monitored by means of *slack value*. This value is the number of generations in which a particular mode of operations is allowed to continue without producing better solution (i.e. association rules with higher fitness value than previously encountered). Once a plateau is reached for more than a specified *slack value*, SARG switches the GP operations to a different mode. Mode 2 alone may be sufficient to move away from the plateau situation encountered in Mode 1. If Mode 2 fails or a similar but new situation in encountered, Mode 3 can then be reinforced. Experiments have shown that Mode 1 is suitable as default initially and oscillation between Mode 2 and Mode 3 achieves the best performance. Figure 4 depicts the self-adjusting evolutionary computation in SARG.

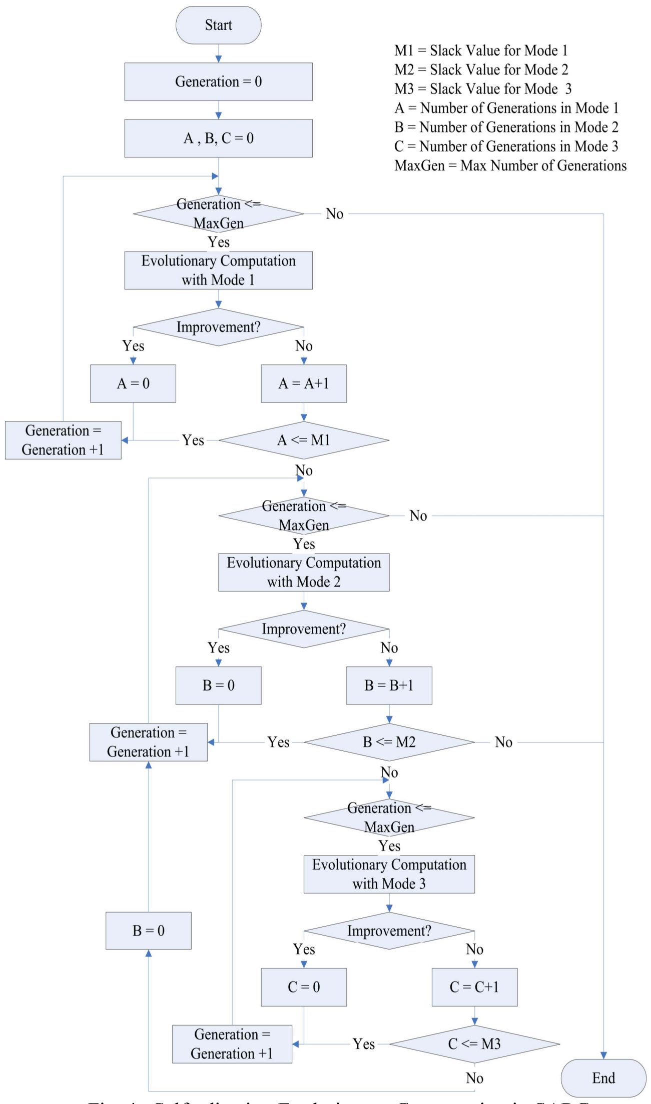
A. Final Rule Builder

Referring to Figure 1, The *Final Rule Builder* in SARG is similar to GPIL. The association rule for each category is nested in the final classification rule in a nested IF..THEN.. ELSE format. Association rules are nested according to the fitness value rather than category number. This is because rules with higher fitness values are likely to classify samples more accurately than those with lower one. The format for the final rule for *n* categories is as follows:

```
IF [condition(s) for the rule with highest fitness value]
THEN (class = category of the rule with highest fitness value)
ELSE  IF [condition(s) for the rule with 2nd highest fitness
          value]
      THEN (class = category of the rule with 2nd highest
            fitness value)
      ……….
      ELSE IF [condition(s) for the rule with lowest fitness
               value]
           THEN (class = category of the rule with lowest
                 fitness value)
           ELSE (sample is unclassified)
```

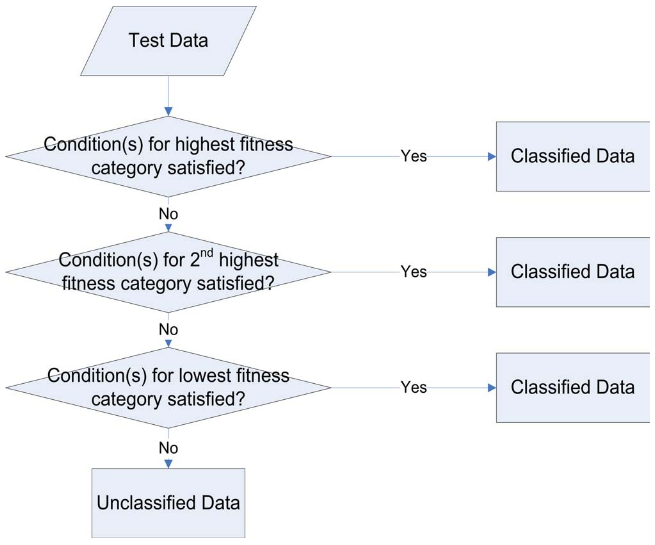
B. Testing

Once the final classification rule has been constructed, it is tested on the test data set to assess the accuracy. The testing process can be described in Figure 5. Classification of a new sample follows this process also.



Start
Generation = 0
A , B, C = 0
M1 = Slack Value for Mode 1
M2 = Slack Value for Mode 2
M3 = Slack Value for Mode 3
A = Number of Generations in Mode 1
B = Number of Generations in Mode 2
C = Number of Generations in Mode 3
MaxGen = Max Number of Generations
Generation <= MaxGen
Yes
No
Evolutionary Computation with Mode 1
Improvement?
Yes
No
A = 0
A = A+1
Generation = Generation +1
Yes
A <= M1
No
Generation <= MaxGen
Yes
No
Evolutionary Computation with Mode 2
Improvement?
Yes
No
B = 0
B = B+1
Generation = Generation +1
Yes
B <= M2
No
No
Generation <= MaxGen
Yes
Evolutionary Computation with Mode 3
B = 0
Improvement?
Yes
No
C = 0
C = C+1
Generation = Generation +1
Yes
C <= M3
No
End

Fig. 4. Self-adjusting Evolutionary Computation in SARG



Test Data
Condition(s) for highest fitness category satisfied?
Yes
Classified Data
No
Condition(s) for 2nd highest fitness category satisfied?
Yes
Classified Data
No
Condition(s) for lowest fitness category satisfied?
Yes
Classified Data
No
Unclassified Data

Fig. 5. Testing and Classification

VI. Classification using SARG

SARG has been tested with benchmark data sets, These sets were the same as used in GPIL for valid comparison.

A. *Datasets used*

Two benchmark data sets which are available in the public domain were selected. These are the IRIS flower classification [17] and the Cleveland heart data [18]. The first is 3-category classification and known to be relatively easy. It is used during the initial verification of SARG. The Cleveland heart data is relatively harder and is of 2-category classification. Another data set selected is the student data at School of Information technology, KMUTT, Thailand. This is the hardest among the three and is of 3-category classification. Their details are shown in Table 2.

TABLE 2.
DATA SETS USED

	IRIS data	Heart Data	Student data
No. of category	3	2	3
No. of attributes in a sample	4	13	8
Total no. of samples	150	297	276
No. of samples in Training set	90	149	200
No. of samples in Test set	60	148	76

B. *Classification Results*

After numerous experiments for the best performance, parametric values for SARG on classification of the three data sets are as summarized in Table 3.

TABLE 3.
SUITABLE PARAMETRIC VALUES

	IRIS data	Heart Data	Student data
Population size	40	40	50
No. of generations	5000	50,000	5000
Duplication rate	25%	25%	25%
Crossover rate	30%	30%	25%
Reproduction rate	25%	25%	25%
Mutation rate	20%	20%	25%
Slack value	30	150	50

Among the above parameters, crossover and reproduction rates are most sensitive.

Performance of SARG is superior to GPIL in terms of both classification accuracy as well as computation efficiency. It yielded higher classification accuracy and was able to find better solutions in shorter time. Comparison in terms of classification accuracy between GPIL and SARG on the three data sets can be summarized in Table 4.

TABLE 4.
SARG V.S. GPIL : CLASSIFICATION PERFORMANCE (ACCURACY)

	IRIS data	Heart Data	Student data
GPIL	96%	83.11 %	53.94%
SARG	100%	87.83 %	55.26%

VII. Discussion

Experiments on the data sets revealed some aspects which must be considered in utilizing evolutionary computation in classification and inductive learning as follows :

Sufficient number of training samples : Enough samples is crucial to the learning process in GP units. Insufficient number of samples will not allow the learning process to discover useful patterns and regularity in the training set. It is better to split the original data set to allow considerably more number in the training set than in the test set.

Similar number of samples in each category : There ought to be similar number of samples in each category in both training and test sets. A training set which is dominated by particular category/categories is detrimental to the learning process. It can be difficult for a GP Unit, which is responsible for category with too few samples, to generate rules with high fitness value. This is because a rule is likely to classify other categories more accurately than the intended one due to higher numbers of samples present. In effect, GP unit is likely to learn what is not rather than what is.

Fitness value in each category : Final Rule Builder in Figure 1 arranges the nested IF.. THEN ..ELSE according to fitness value instead of category number to achieve optimum performance. This presumes that a rule with reasonably high fitness value is attained for each category. If GP units can only obtain rules with low fitness values, this indicates that the classification may be inherently too hard.

Suitable fitness function : While the importance of this is commonly understood in evolutionary computation, the same can be said about SARG and should not be overlooked. This may likely be application dependent.

Suitable slack value : Performance, especially in terms of computation efficiency, of SARG depends on setting appropriate *slack value*. This study, so far, indicated that optimum slack value may also depend on the maximum size of tree (i.e. number of nodes) allowed as well as number of attributes under consideration.

VIII. Conclusion and Future Work

While there have been several applications of evolutionary computation in inductive learning, the work described in this paper is yet another attempt to apply Evolutionary Computation in inductive learning for classification. Ability to adjust the evolutionary computation to suit the task at hand is advantageous and can rectify lack of progress similar to those in plateau situations. This work presents a MaxToMin crossover for genetic programming and Self-adjusting selection. MaxToMin crossover may improve the chance of producing better chromosomes while Self-adjusting selection provides an alternative path to reproducing new chromosomes in situations where no progress is possible after successive number of generations.

Future work can be carried out in the following areas:

Generic fitness function : As fitness function is critical to the rules generation in GP units. While a routine to produce a tailor made fitness function for every data set may not be possible, a generic fitness function may be possible. A routine can then be implemented to determine optimal parametric values for the generic fitness function.

Methods used for Final Rule Builder : The nested IF.. THEN.. ELSE for the final rule is ordered according to fitness value at present. Further investigation can be done on ordering the final rule according to other properties. Candidates for this are number of conditions in the rules and number of samples in each category.

Optimal slack value determination : A suitable slack value is likely to depend on the nature of the data set. The study so far has indicated that two factors, maximum tree size (i.e. no. of maximum nodes) allowed and number of attributes in a sample, have an influence on the performance of each GP unit. These two factors are strong candidates which merit further analysis.

Acknowledgment

The author gratefully acknowledges the technical advice from Dr. R. Alcock. He also wishes to acknowledge Mr. Ekarat Rattagan for his contribution and assistance throughout the project. The support for computing facilities offered by School of Information Technology at King Mongkut's University of Technology Thonburi is appreciated. Preparation of this paper would not have been possible without the help and moral support from his wife, Dr. P. Lavangnanada.

References

[1] R. Kennedy, e. al., *Solving Data Mining Problems Through Pattern Recognition,* Prentice Hall, New Jersey, USA, 488 p., 1998.

[2] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules", In *Proceedings of the 20th Intl. Conf. On Very Large Databases*, Santiago, Chile, September 1994.

[3] T. M. Mitchell, *Machine Learning*, McGraw-Hill, Massachusetts, USA., 414 P., 1997.

[4] *W. Banzhaf, e. al., Genetic Programming : An Introduction,* Morgan Kaufmann, California, USA ., 450 p., 1998.

[5] J. R. Quinlan, "Induction of decision trees", *Machine Learning*, *Vol. 1, No. 1,* pp. 81 - 106, 1986.

[6] J. R. Quinlan, *C4.5: Programs for Machine Learning,* Morgan Kaufmann, San Mateo, USA, 1993.

[7] Breiman, et al., *Classification and regression trees,* Wadsworth International Group, Belmont, California, USA, 1984.

[8] P. Povalej, et al., "Classic Decision Tree Induction Versus Genetic Algorithms", In *Proceedings of 4th Asia-Pacific Conference on Simulated Evolution And Learning (SEAL'02), 18-22 November, Singapore, 2002.*

[9] W. Spears and K. A. DeJong, "Using Genetic Algorithms for Supervised Concept Learning", In *Proceedings. of IEEE Conference on Tools for AI,* Washington DC, USA, pp. 335-341, 1990.

[10] K. A. DeJong, W. Spears and D,F. Gordon, "Using Genetic Algorithms for Concept Learning", *Machine Learning, Vol. 13,* pp. 161-188, 1993.

[11] C. C. Bojarczuk, et al., "A constrained-syntax genetic programming system for discovering classification rules: application to medical data sets", *Artificial Intelligence in Medicine, Vol. 30*, pp.27-48, 2004.

[12] M. Khabzaoui, C. Dhaenens and E-G. Talbi, "Parallel Genetic Algorithms for multi-objective rule mining", In *Proceedings of 6th Metaheuristic International Conference (MIC2005)*, pp. –179-188, 22nd-26th August, Vienna, Austria, 2005.

[13] A. Ghosh and A. A. Freita (Eds.), *Special issue on data mining and knowledge discovery with evolutionary algorithms, IEEE Trans. on Evolutionary Computation, Vol. 7, No. 6*, pp. 517-575, 2003.

[14] A. A. Freitas, *Data Mining and Knowledge Discovery with Evolutionary Algorightms,* Springer-Verlag, 262 p., 2002.

[15] K. Lavangnananda, "A Genetic Programming Approach to Inductive Learning", In *Proceedings of the Int. Conf. Computational Intelligence for Modelling, Control and Automation (CIMCA'2004)*, 12th – 14th July, Gold Coast, Australia, 2004.

[16] R. J. Alcock and Manolopoulos, "Using Genetic Algorithms for Inductive Learning", In *Proceedings of 3rd Int. Multiconference on Circuit, Systems, Communications and Computers (CSCC'99*), Athens, Greece, 4th – 7th July, 1999.

[17] R. A. Fisher, *"The Use of Multiple Measurements in Taxonomic Problem"s, Annual Eugines, Vol. 7, Part 2,* pp. –179-188, 1936.

[18] C. Blake, E. Keogh and C. J. Merz, *UCI Repository of Machine Learning Databases,* Irvine, CA : University of California, Dept. of Information and Computer Science, [http://www.ics.uci.edu/~mlearn/MLRepository.html], 1998.

[19] K. Lavangnananda, "Synergistic Genetic Algorithm for Inductive Learning (SynGAIL)", In *Proceedings of the Int. ICSC Congress on Computational Intelligence : Methods and Applications (CIMA'2001*), pp. 443-449, 19-22 June, University of Wales Bangor, U.K., 2001.

[20] S. J. Russell and P. Norvig, *Artificial Intelligence A modern approach, 2nd Edition*, Prentice-Hall International, Inc., USA., 2003.

[21] A. Kamiya, et al., "Worker Ants' Rule-Based Genetic Algorithms Dealing with Changing Environments", In *Proceedings of 2005 IEEE Mid-Summer Workshop on Soft Computing in Industrial Applications (SMCia/05)*, pp. 117 – 121, 28th – 30th June, Helsinki University of Technology, Espoo, Finland, 2005.

[22] J. Martikainen and S. J. Ovaska, "Hierarchical Two-Population Genetic Algorithm", In *Proceedings of 2005 IEEE Mid-Summer Workshop on Soft Computing in Industrial Applications (SMCia/05)*, pp. 91 – 98, 28th – 30th June, Helsinki University of Technology, Espoo, Finland, 2005.