Project Design

Mengjun Xu
Supervisor: Prof. Steven Guan
Department of Computer Science and Software Engineering
Xi'an Jiaotong-Liverpool University

1 SUMMERY OF PROPOSAL

1.1 Background and Related Works

With the boosting quantity of data in the information age, data mining has become one of the most prevailing computer science research topics in recent decades, among which subfield pattern classification is the most significant problem concerning with various real life applications such as speech recognition and image processing.

The core idea of pattern classification is to recognize the category of a new item. Due to a wide range of applications, numerous algorithms have been established to pursue an accurate, efficient or relatively universal method to address the classification; for example, Naïve Bayes classifier[1], fuzzy logic[2][3], Support Vector Machine (SVM)[4], Neural Networks (NN)[5], and Genetic Algorithms (GAs)[6].

This project will focus on Genetic Algorithms, an embranchment of Evolutionary Computation, which was initially proposed by J.H. Holland in 1975[6]. Rule-based Genetic Algorithms are one of the most successful and commonly used GA approaches among which Pittsburgh approach and Michigan approach are two main branches that uses different number of chromosomes as the classification rule(s) [7][8][9]. However, there is still substantial potential for GA-based classifiers to increase in both accuracy and efficiency. Fang and Guan[10] have design the Recursive Learning of Genetic Algorithm with Task Decomposition and Varied Rule Set (RLGA) to train the given data set recursively by decomposing the original classification problems into sub-problems. The experiment results of this novel idea turned out to have a significant improvement in data sets under a dimension of twenty compared to the conventional GAs.

However, RLGA loses its superiority as the dimension gets higher. To enhance the competence of RLGA, Recursive Learning of Genetic Algorithm Featuring Incremental Attribute Learning (RLGA-IAL)[11] was proposed to cope with higher dimensional data classification. Instead of training attribute in a batch manner, RLGA-IAL integrates attributes sequentially. This linear approach results in a successful and

promising performance in solving datasets with a dimension ranging from 5 to 50.

Although RLGA-IAL does solve the problem with a dimension reaching 60, it suffers from considerable training time plus relatively low accuracy rate. Therefore, a parallel style of RLGA-IAL, named as **Recursive Learning Genetic Algorithm - Parallel Incremental Learning (RLGA-PIL)**, is proposed in this project to address these drawbacks. RLGA-PIL will be focusing on concurrent manner of classification that deals with several numbers of attribute parallelly to increase the efficiency compared with RLGA-IAL. In addition to this, a preferable way of grouping attributes is to be found to enhance the accuracy of the new proposed algorithm.

1.2 Changes against specification

In the project specification, we mainly intended to focus on converting the sequential manner of learning in RLGA-IAL into different parallel manners by training two or more attributes at a time in a tree structure. And the original purpose of this project is to further explore the potential of RLGA-IAL to find a new genetic algorithm to address the previously mentioned drawbacks and to broaden the algorithm's application by applying it to other relevant computer science branches study such as regression if possible. After this month's exploration, there are several modifications.

The core exploring field of the RLGA-PIL is changed. The new objective of the project will be focusing on how to divide the attributes into different groups rather than the number of attributes in each group. The reason of this variance is quite intuitively understandable as the accuracy of a classifier is much more significant than the efficiency. Have a probe into a variety of tree structures for concurrent execution will certainly have impacts on the performance of RLGA-PIL; but the influences will mainly on the speed of the execution but rarely on the accuracy. In order to seek for a preferable algorithm that leads to relatively satisfactory classification accuracy, explorations should be mainly done on the methods of dividing attributes into different groups for a classification problem. In this case, the parallel structure of integrating rules for each attribute will be fixed as a binary tree when exploring the influences among attributes.

1.3 Progression So Far

Research review work has almost been finished since the project specification. In the recent months, we mainly focused on the design and implementation of the newly proposed genetic algorithm which is RLGA-PIL. In order to have a fully understanding of the previous two algorithms (RLGA and RLGA-IAL), large amount of experiments have been conducted to check the consistency of implementation results of the two algorithms compared with the results in corresponding papers. Various parameters settings of two algorithms have been tested as well.

Considering one of the major tasks in this project is converting the original RLGA-IAL into parallel RLGA-PIL, programming on concurrent programming and Genetic Algorithms implementations are done to ensure the quality of the algorithms and codes. Last but not least, the implementation of RLGA-PIL has been partially fulfilled to prepare for the following prototyping.

2 DESIGN OF THE PROPOSED ALGORITHM

2.1 Algorithm Overview

The general idea of the new proposed algorithm RLGA-PIL is inspired by the time-consuming training of previous RLGA-IAL. In a pattern recognition problem, incremental training usually takes much more time than batch training. In order to weaken this drawback and enhance the accuracy of RLGA-IAL, instead of picking up one attribute at a time for training, we are going to pick up two attributes at a time.

In the RLGA-IAL, attributes are picked one at a time according to its discriminating ability[11]. The stronger discriminating ability an attribute has, the higher training priority this attribute has and can be integrated into the final classification rule earlier. However, rule learnt through a single attributes along with its label may lead to skewing results with not enough circumstances taken into consideration. From this point of view, we are going to involve two attribute each time. Furthermore, for most classification problems, especially for those have a high dimension, there exist various relationships between attributes which might be either similarities or differences. Closely related attributes usually have more common in classification results and can help each other in the training section whereas others may cause interferes. Therefore, currently the idea is to train similar attributes together; and if such attribute does not exist, we will try to avoid interferes. The measurement of "similarity" will be discussed later in the following section.

Theoretically, this variation can not only accelerate the algorithm to some degree but also reduce the misclassification rate compared with previous proposed GAs. Of course, the actual results of RLGA-PIL are to be tested and analysed later on.

2.2 Genetic Algorithms

Genetic algorithm, which imitates the evolution process of nature organisms, almost follows the same flow as Evolution Algorithm (EA). Starting from the initial status of a population, the updated population will keep selecting ones who have the better fitness to generate the next generation through crossover and mutation until the expected condition is satisfied. Here the fitness of an instance represents the applicability of the rule for the classification problem. The higher fitness a rule have, more useful it will be. Here is the pseudocode for canonical Rule-based Genetic

Algorithm.

Algorithm1. Rule-based Genetic Algorithm

t = 0;
Initialise a population *P*(0);
Evaluate(*P*(0));
WHILE the terminal condition is NOT satisfied **DO**
    P'(*t*) = *Select*(*P*(*t*));
    P'(*t*) = *Crossover*(*P*(*t*));
    P'(*t*) = *Mutate*(*P*(*t*));
    Evaluate(*P*(*t*));
    P(*t* + 1) = *Combine*(*P*(*t*), *P*'(*t*));
    t = *t* + 1;
END

2.3 GA classifier

In order to construct a classifier for a specific classification problem by genetic algorithm, we need to encode rules into a set of IF-THEN digital codes to represent the problem. Each rule is considered as a chromosome as a member of the whole population. Usually, the original data obtained may not consist of numbers only. Therefore, preprocessing need to be conducted for further operations. For example, various class names are encoded into integers; incomplete information (e.g. an attribute of value NAN) will be eliminated from this instance etc.

Here is the general format of a GA classifier where every three elements ($\mathrm{Act_i\ MIN_i\ MAX_i}$) of the chromosome represent a specific gene for attribute $\mathrm{A_i}$:

$\mathrm{Act_1\ MIN_1\ MAX_1}$	$\mathrm{Act_2\ MIN_2\ MAX_2}$	$\mathrm{Act_3\ MIN_3\ MAX_3}$	…	$\mathrm{Act_n\ MIN_n\ MAX_n}$	CLASSNO

where

$\mathrm{Act_1}$, $\mathrm{Act_2}$…$\mathrm{Act_n}$ denote the status of attribute $\mathrm{A_i}$, either active(1) or inactive(0);
$\mathrm{A_1}$, $\mathrm{A_2}$…$\mathrm{A_n}$ denote *n* attributes;
$\mathrm{MIN_i}$ denotes the lower bound for attribute $\mathrm{A_i}$ where $i \in [1, n]$;
$\mathrm{MAX_i}$ denotes the upper bound for attribute $\mathrm{A_i}$ where $i \in [1, n]$;
CLASSNO denotes the conclusion for classification which specifies the category.

Hence, the whole rule tells:

IF $(\mathrm{A_1} \in [\mathrm{MIN_1},\ \mathrm{MAX_1}]) \wedge (\mathrm{A_2} \in [\mathrm{MIN_2},\ \mathrm{MAX_2}]) \wedge \ldots \wedge (\mathrm{A_n} \in [\mathrm{MIN_n},\ \mathrm{MAX_n}]))$ THEN (this instance belongs to class CLASSNO).

Consequently, the problem of treat GAs as a classifier is actually the problem of finding an optimal rule to tell the class.

2.4 Genetic operators

There are three main operations in a genetic algorithm which are selection, mutation and crossover.

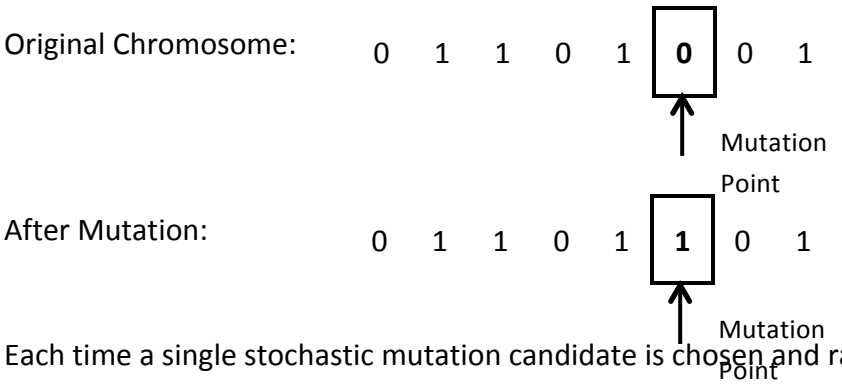
Selection

Since genetic algorithm mimics the nature selection, there are parents and offspring in a population which is composed of a set of chromosomes. In this project, Elitist Strategy[12] will be applied first to prevent from losing the best rule at a training iteration. Then, we apply fitness proportionate selection, commonly known as Roulette wheel selection rule[13], to select parents for reproduction. Therefore, the fitness of a rule determines whether a rule can be kept or not. Rules with higher fitness are more likely to be copied or reproduced into the next generation.

The detailed measurement of fitness of a rule (i.e. chromosome) will be discussed in details in the following section.

Mutation

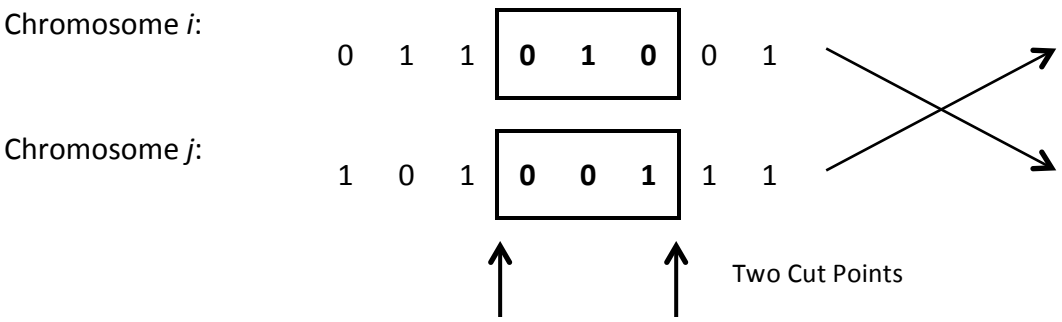
Similar to the gene mutation in biology model, mutation in genetic algorithms is a fairly rare occurrence. The mutation rate is commonly set less than 1% to make sure the stability of a population. The following diagram shows how a mutation works in a chromosome.



Each time a single stochastic mutation candidate is chosen and randomly changed.

Crossover

The crossover operation in a genetic algorithm imitates the gene development in a biological organism as well. This operation is much more active compared to mutation; usually with a frequency of occurrence around 80%.



After Crossover:

Chromosome *i'*: 0 1 1 **0 0 1** 0 1

Chromosome *j'*: 1 0 1 **0 1 0** 1 1

Same as mutation, the position for conducting crossover is random each time.

2.5 Fitness, Coverage and Similarity

Fitness function plays a significant role in genetic algorithm. The function chosen directly determines the quality of training results. Now that the primary concern of this project is the accuracy of the classification results, we will utilize a measurement regarding to the accuracy. Following is one of the most commonly used fitness function where '#' denotes 'number'. To distinguish this conventional measurement with the to-be discussed new concept, we name this fitness function as global fitness.

$$f_{Global} = \frac{CorrentlyClassifiedPattern\#}{TotalPattern\#}$$

RLGA proposed a novel idea of local fitness to measure the fitness of a chromosome:

$$f_{Local} = \frac{CorrentlyClassifiedPattern\#}{ApplicablePattern\#}$$

To assist the local fitness function, we also utilize idea of coverage:

$$c = \frac{ApplicablePattern\#}{TotalPattern\#}$$

Apart from the above functions applied in previous algorithms, we devise a new concept of 'Similarity' for grouping the attributes. Here is a preliminary idea of calculating the degree of correlation between two attributes details to be discussed in the following section:

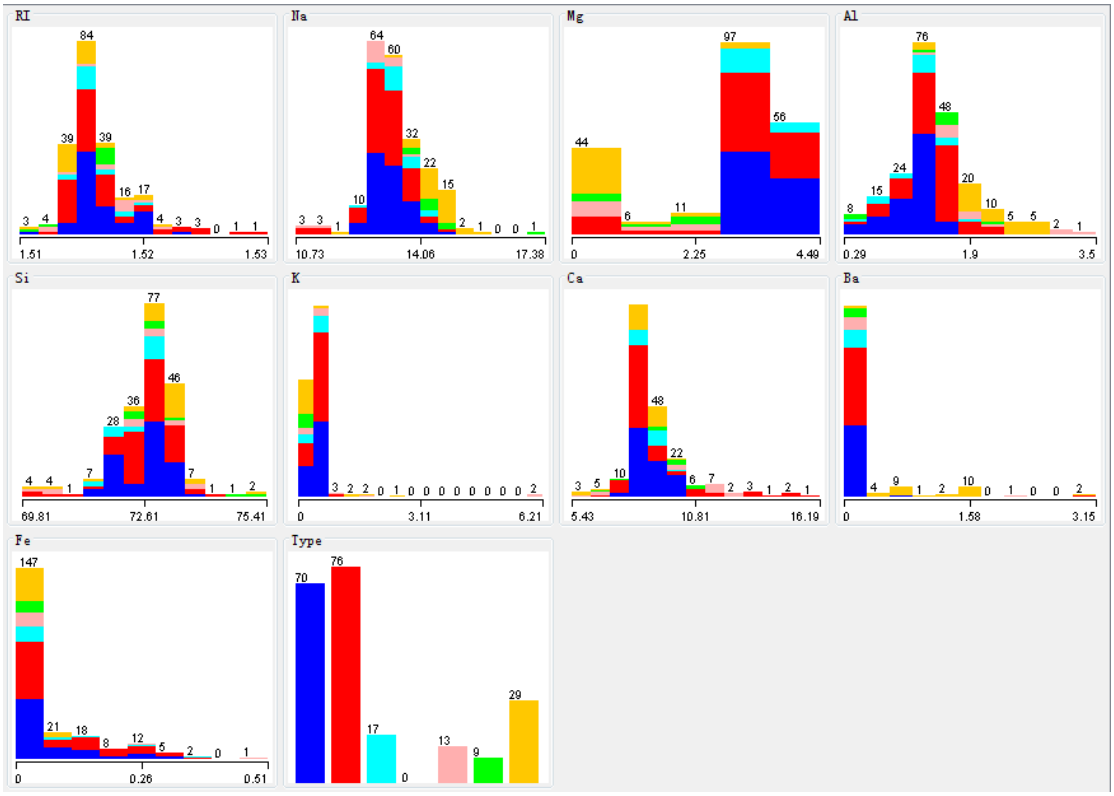
$$s(A_i, A_j) = \text{ClassDistributionSimilarity}(A_i, A_j)$$

Given a specific attribute A_i, the best pair candidate should have the largest similarity among all possible pairs.

$$\text{Best pair's similarity } = max_{i=0\ \&\ i\neq j}^{n}\{s(A_i, A_j)\}$$

The actual implementation and applicability of this Similarity function is to be tested and verified in the later experiments and will be compared with random grouping and other grouping methods. Therefore, this function may be improved later on. In addition, artificial grouping, in-order grouping and random grouping may be tested

as well to see the performance comparing with the automatic grouping here.



Intuitively, take dataset Glass from UCI database[14] for instance. The above graph shows the class distribution of each attribute. The idea is attributes with a similar class distribution share more in common. In this graph, attributes RI, Na, Al, Si and Ca shows the similarity property mentioned just now, and can be grouped together.

2.6 Training Phase

Since this project is a twist of the RLGA and RLGA-IAL, the general process of the training algorithm in RLGA-PIL is designed based on these two algorithms. It makes up of two sections, Decomposition Training (D-Train) and Global Control.

Algorithm2. Decomposition Training Algorithm

$t = 0$;
Initialise a set of chromosomes $P(0)$;
Evaluate the local fitness and coverage of ($P(0)$);
WHILE the terminal condition is NOT satisfied **DO**
    $P'(t) = Select(P(t))$;
    $P'(t) = Crossover(P(t))$;
    $P'(t) = Mutate(P(t))$;
    Evaluate the local fitness and coverage of ($P(t)$);
    $P(t + 1) = Combine(P(t), P'(t))$;
    $t = t + 1$;
END

The decomposition training follows the general process of a genetic algorithm. However, in the evaluation part, we apply the local fitness along with the coverage instead of the traditional global fitness function.

The Global control section is in charge of the whole RLGA-parallel incremental learning. The most complicated difference in this global control algorithm compared with RLGA and RLGA-IAL is that it involves the grouping work.

Algorithm3. Global Control Algorithm

Input: Training set D_{Train}, expert array Exp, original attribute order $AttrOrder$.
Output: Updated expert array Exp.

FOR each unmarked attribute A_i **DO**
    Find the most similar attribute through *Similarity Evaluation Algorithm*
    Apply Decomposition Training Algorithm on this attribute pair
    Mark both attributes in this pair
END

Here is the pseudocode for the algorithm that evaluates the similarities between the attributes based on variance. However, this is currently just an incipient theoretical idea of judging the similarities and may be improved in the prototyping section later.

Algorithm4. Similarity Evaluation Algorithm

Input: Training set D_{Train}.
Output: Pairing result.

Initialise an array $Variance_i[]$ for each attribute with a dimension equals to the number of class.
FOR *j*th class **DO**
    FOR each attribute A_i **DO**
        $Variance_i[j]$ ← Calculating the variance of the attribute value
    END
END
n = AttrNo;
FOR attribute A_i **DO**
    tempMax = 0;
    bestPair(*i*) = -1;
    WHILE $\mathrm{i}+1 \leq \mathrm{n}$ **DO**
        $\mathrm{j} = \mathrm{i}+1;$
        $Difference_{ij} = |\, Variance_i - Variance_j \,|.$
        d = *sum*($Difference_{ij}$).
        IF d>*tempMax* **THEN**
            tempMax = *d*;
            bestPair(*i*)=*j;*
        END IF
    END
END

2.7 Integration Phase

Integration is one of the challenges in this project and is fairly significant to RLGA-PIL due to its idea of decomposition problem solving. Rules for attributes are learnt group by group which means the integration work is more than simply unite experts for each group into a huge expert. More importantly, the integration section has the responsibility to give an expert that includes all the rules for each attribute that learnt in the training section and arrange them in order.

When integrating the rules for attributes, we will just integrate in the order that a rule is generated first. Each rule for an attribute is labeled with its original attribute number, so that after all the integrations done, the rule then will be reorganized into the original attribute order for easy testing. The general idea can be seen in the following diagram along with the pseudocode in Algorithm5.

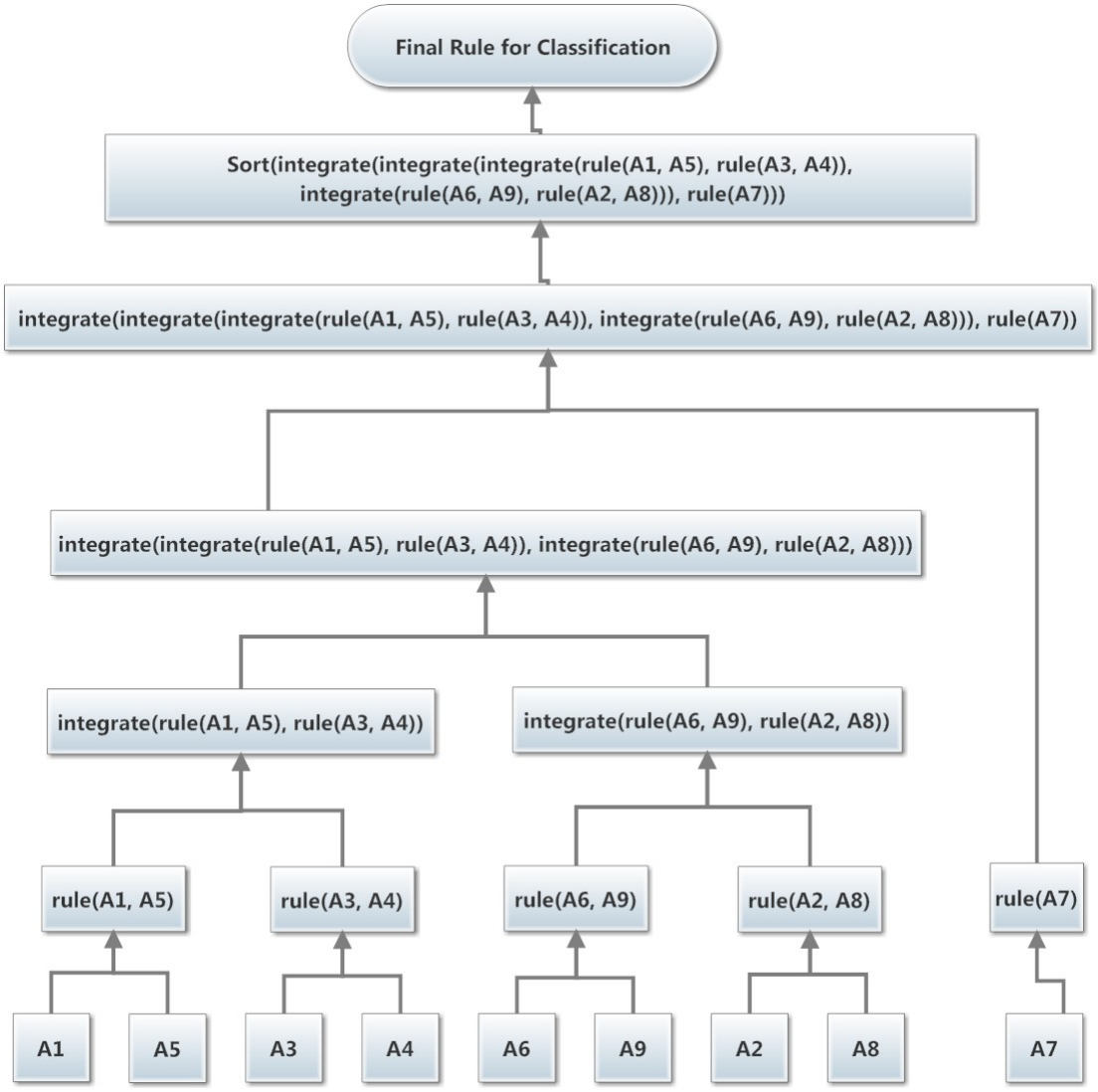


Figure 1. Integration process of RLGA-PIL

Algorithm5. Integration Algorithm

Input: New experts.
Output: The new integrated expert.

FOR each incoming new experts **DO**
Append the new expert to the previous new integrated expert;
END
Rearrange the order of the rules based on original attribute label.

3 EVALUATION DESIGN

3.1 Evaluation Method

The method we are going to apply in this project is 10-fold cross validation.

There are several categories of cross validations, among which holdout method is the simplest. The holdout method split the original dataset into two parts randomly; one part as a training set while the other part as a testing set. Due to the randomness of the data partition, the experiment results will rely on the partition result to a large extent which leads to much variation.

Consider this fluctuation, K-fold cross validation method will be used here. K-fold cross validation can be regarded as an extension of the holdout method. It is one of the statistical techniques that can provide a relative stable and reliable model. In addition, this method can provide a relatively precise average result considering the number of training and testing it done. The variable '*K*' here stands for the number of subgroups divided from the original dataset. Each subgroup will be chosen to be the validation data once with remaining (*K*-1) subgroups are the training data. Consequently, the final outputs will be calculated out of *K* sub-experiments.

The evaluation results will be compared with ones generated by canonical GAs, RLGA, RLGA-IAL and probably other relevant algorithms to explore the strengths and vulnerabilities of RLGA-PIL.

3.2 Benchmark Datasets

Data for experiments will be picked from UCI Repository of Machine Learning[14]. Details please refer to the following table:

Dataset Name	Attribute Number	Class Number	Instance Number
Iris	4	3	150
Yeast	9	10	1484
Glass	9	6	214
Cancer	9	7	214
Wine	13	3	178
Label	16	2	57
Zoo	17	7	101
Lymphography	18	4	148
Segment	19	7	2310
Horse Colic	27	2	368
Ionosphere	34	2	351
Anneal	38	6	898
SPECTF Heart	44	2	267
Lung Cancer	56	3	32
Sonar	60	2	208

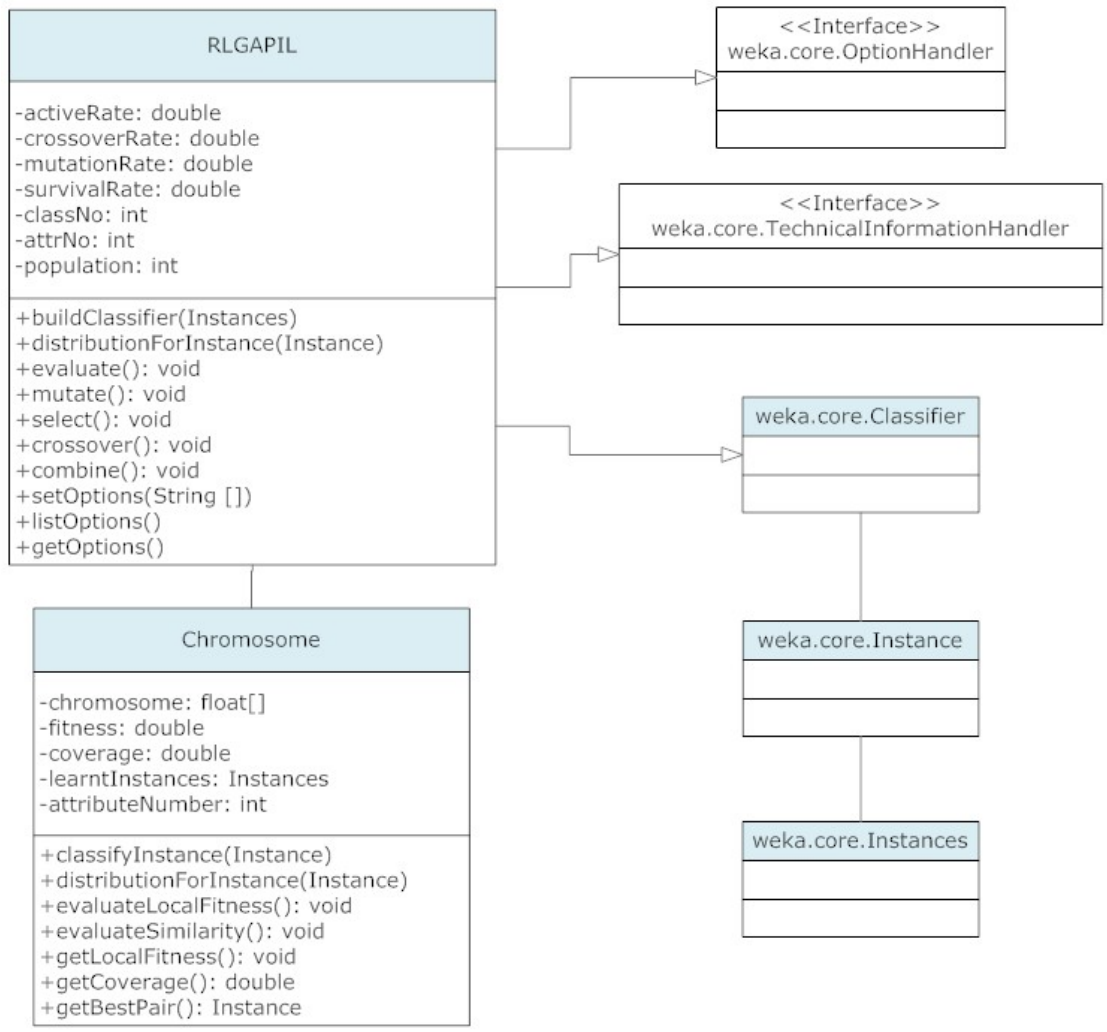


Figure 2. Class Diagram for RLGA-PIL

4 IMPLEMENTATION DETAIL

The implementation of this project will be mainly conducted in Weka[15], one of the most prevailing free software written in Java for dealing with problems in machine learning field. It includes a large amount of commonly used classical algorithms. More importantly, it allows us to develop our own algorithm as well.

In addition, Weka provides a total package of Graphical User Interface. Therefore, this project will directly utilize the interface provided.

To have a look into the class diagram for the project, please refer to Figure2.

5 REVIEW AGAINST PLAN

Generally, the progress of this project follows the planning we made in the project specification. Tasks in this month's schedule have been done and their status has been updated in schedule table and corresponding Gantt chart. To see more details, please refer to the Appendix.

REFERENCES:

[1] M. Pelikan, D. E. Goldberg, and E. Cantú-paz, "Linkage problem, distribution estimation and Bayesian networks," *Evol. Comput.*, vol. 8, no., 3, pp. 311-340, 2000.

[2] H. Ishibuchi, T. Nakashima and T. Murata, "Performance evaluation of fuzzy classifier systems for multidimensional pattern classification problems", *IEEE Trans. Syst., Man, and Cybern.,* pt. B, vol.29, no. 5, pp.601-618, 2002

[3] H. Ishibuchi, T. Nakashima, "Improving the performance of fuzzy classifier systems for pattern classification problems with continuous attributes," *IEEE Trans. Ind. Electron.*, vol. 46, no. 6, pp. 1057-1068, Dec. 1999.

[4] B. Schölkopf, C.J.C. Burges, A.J. Smola, *Advances in Kernel Methods: Support Vector Learning*. MIT Press, 1999.

[5] R. Polikar, L. Udpa, and V. Honavar, "Learn ++: An incremental learning algorithm for supervised neural networks," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, pt. C, vol. 31, no. 4, pp. 497-508, Nov. 2001.

[6] J. H. Holland, *Adaptation in Natural and Artificial System*, Univ. of Michigan Press, 1975.

[7] O. Cordon, F. Herrera et al, "Evolutionary Computation," in *Genetic Fuzzy Syst.: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases*. Singapore: *World Scientific*, vol. 19, 2001, pp. 47-78.

[8] K. D. Jong, "Learning with genetic algorithms: An overview," in *Mach. Learning*, vol. 3, no. 2, pp. 121-138.

[9] M. Setness and H. Roubos, "GA-fuzzy modeling and classification: complexity and performance," in *IEEE Trans. Fuzzy Syst.*, vol. 8, no. 5, pp. 509-522.

[10] L. Fang, S. U. Guan and H. F. Zhang, "Recursive Learning of Genetic Algorithms with Task Decomposition and Varied Rule Set", in *Int. J. of Ap. Evol. Computation (IJAEC)*, vol.2 No.4, 2011.

[11] H. F. Zhang, L. Fang and S. U. Guan, "Recursive and Incremental Learning GA Featuring Problem-Dependent Rule-Set," in *Bio-Inspired Computing and Applicat., LNBI*, vol.6840, pp. 215-222, 2012.

[12] Q. B. Zhang, T. H. Wu and B. Liu, "A Population-Based Incremental Learning Algorithm with Elitist Strategy," in *Natural Computation, 2007. Third International Conference on*, vol.3, no., pp.583-587, 24-27 Aug. 2007

[13] S. U. Guan and K. Ramanathan, "Percentage based hybrid training with Neural Network Specific Crossover," in *Intelligent Systems*, vol. 16, no. 1, pp. 1-26, 2007.

[14] C. L. Blake and C. J. Merz. (1998) UCI Repository of Machine Learning Databases. Dept. Inform. and Comput. Sci., Univ. of California, Irvine. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html.

[15] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. H. Witten, "The WEKA Data Mining Software: An Update", in *SIGKDD Explorations*, Volume 11, Issue 1, 2009.

APPENDIX:

Item#	Task	Duration	Start Date	Finish date	Complete %	Predecessors
1	Literature Review	26	12-09-03	12-09-28	100%	
2	State-of-art Survey	25	12-09-17	12-10-11	100%	
3	Project Plan & Risk Analysis	15	12-10-12	12-10-26	100%	
4	Project Specification	0	12-10-26	12-10-26	100%	
5	Implementation of RLGA	10	12-10-27	12-11-05	100%	
6	Implementation of RLGA-IAL	10	12-11-06	12-11-15	100%	
7	Prototyping	21	12-11-16	12-12-06	80%	
8	Design document	0	12-12-07	12-12-07	100%	
9	Prototyping & Evaluation	9	12-12-07	12-12-15	0%	7
10	Continue prototyping	24	13-01-03	13-01-26	0%	
11	Evaluation & Refinement	33	13-01-27	13-02-28	0%	10
12	Testing	23	13-03-01	13-03-23	0%	11
13	Final Implementation	18	13-03-24	13-04-10	0%	12
14	Experiment & Data Collection	21	13-04-11	13-05-01	0%	13
15	Result Analysis	20	13-05-02	13-05-21	0%	14
16	Dissertation	0	13-05-24	13-05-24	0%	
17	Preparing for presentation	7	13-05-25	13-05-31	0%	
18	Presentation	0	13-05-31	13-05-31	0%	

Table. Project schedule

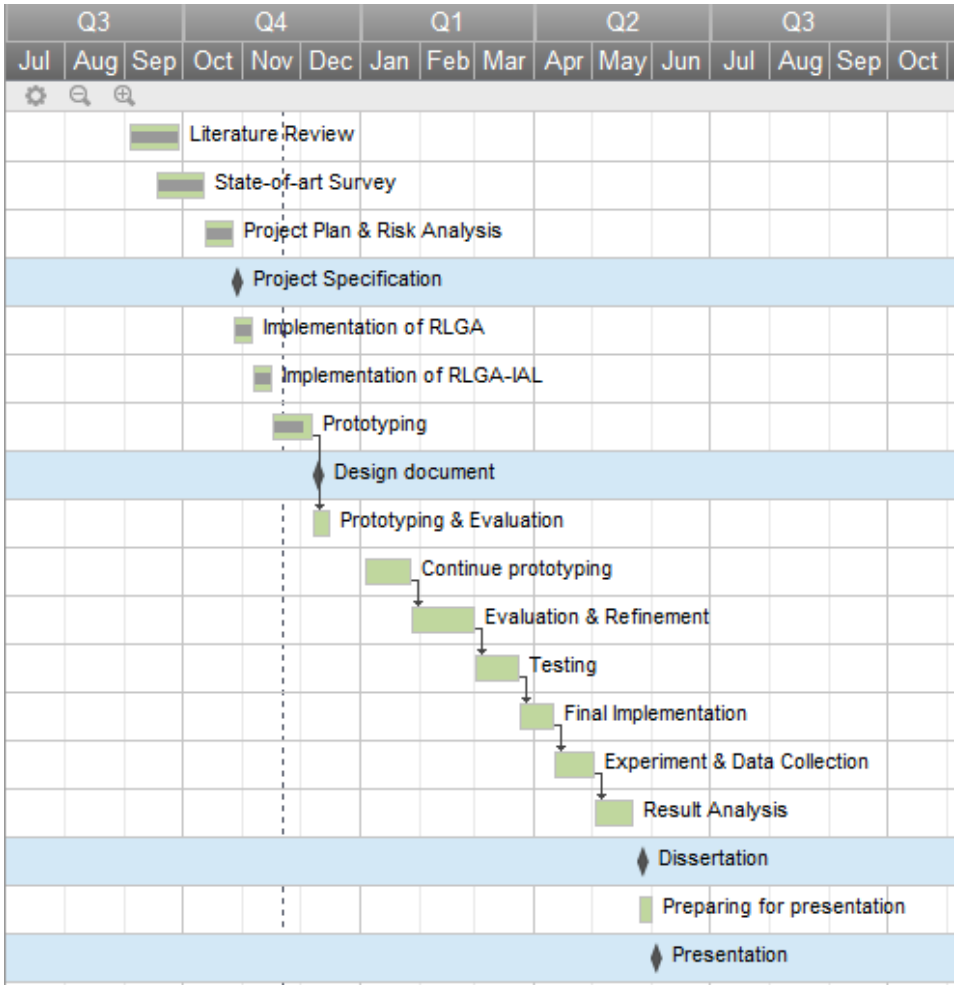


Figure. Gantt Chart for Project schedule