An Incremental Approach to Genetic-Algorithms-Based Classification

Sheng-Uei Guan and Fangming Zhu

***Abstract*—Incremental learning has been widely addressed in the machine learning literature to cope with learning tasks where the learning environment is ever changing or training samples become available over time. However, most research work explores incremental learning with statistical algorithms or neural networks, rather than evolutionary algorithms. The work in this paper employs genetic algorithms (GAs) as basic learning algorithms for incremental learning within one or more classifier agents in a multiagent environment. Four new approaches with different initialization schemes are proposed. They keep the old solutions and use an "integration" operation to integrate them with new elements to accommodate new attributes, while biased mutation and crossover operations are adopted to further evolve a reinforced solution. The simulation results on benchmark classification data sets show that the proposed approaches can deal with the arrival of new input attributes and integrate them with the original input space. It is also shown that the proposed approaches can be successfully used for incremental learning and improve classification rates as compared to the retraining GA. Possible applications for continuous incremental training and feature selection are also discussed.**

***Index Terms*—Classifier agents, genetic algorithms (GAs), incremental learning.**

I. Introduction

TRADITIONAL machine learning techniques have focused on nonincremental learning tasks. It is assumed that the problem to be solved is fixed and the training set is constructed *a priori*, so the learning algorithm stops when the training set is fully processed. On the other hand, incremental learning is an *ad hoc* learning technique whereby learning occurs with the arrival of new data rather than from a fixed set of data, i.e., it is a continuing process rather than a one-shot experience. Incremental learning is more suitable for software agents, as most learning tasks for agents are incremental and agents must adapt to the ever-changing environment incrementally and continuously [1], [2].

Pattern classification problems have been widely used as traditional formulation of machine learning problems and researched with different machine learning approaches including statistical methods [3], neural networks [4]–[6], genetic algorithms [7], [8], fuzzy sets [9], [10], and cellular automata [11]. In recent years, hybrid approaches are also emerging, as they are better in solving more complicated classification problems [12]. We use classification problems as our research vehicle for incremental learning, and call software entities that have the ability to undertake certain classification tasks as classifier agents.

Incremental learning has attracted much research effort in the literature. However, most work explores incremental learning with statistical algorithms [3] or neural networks [4], and none touches on the use of evolutionary algorithms. This paper employs genetic algorithms (GAs) as basic learning algorithms for incremental learning.

Evolutionary algorithms, such as evolutionary programming [13], genetic programming [14], evolution strategies [15], and genetic algorithms [16], have attracted much interest in various applications. They have been shown to be effective in exploring large and complex spaces in an adaptive way, which reflects some of the key aspects of natural evolution mechanisms such as reproduction, crossover, and mutation. Among them, GAs have been widely used in the literature as an evolutionary way to learn classification rules, either through supervised or unsupervised learning [17].

Most work in the literature focuses on batch-mode, static domain, where the attributes, classes and training data are all determined in advance and the task of the GAs is to find out the best rule set which classifies the available instances with the lowest error rate [18]. However, some learning tasks do not fit into this static model. As the real-world situation is more dynamic and keeps changing, a classifier agent is actually exposed to a changing environment. Therefore, it needs to evolve its solution to adapt to various changes. In general, there are three types of changes in classification problems. First, new training data may be available for the solution to be refined. Second, new attributes may be found to be possible contributors for the classification problem. Third, new classes may become possible categories for classification. To deal with these types of changes, classifier agents have to learn incrementally and adapt to the new environment gradually. This paper chooses the arrival of new attributes as the target for incremental learning.

To achieve incremental learning, the standard way in which GAs are applied can be revised. With a scenario of new attributes being acquired, a classifier agent needs some algorithms to revise its rule set to accommodate the new attributes. That means it should find out how new attributes can be integrated into the old rule set to generate new solutions. Of course, the agent can run GAs from scratch again as some conventional approaches do. However, this approach requires a lot of time and wastes the previous training effort. In some real-time applications, there are hard constraints on time and resource; thus, a classifier agent may need to respond quickly in an online manner. For example,

Manuscript received April 5, 2004; revised August 7, 2004 and December 6, 2004. This paper was recommended by Associate Editor H. Takagi.

The authors are with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore 119260 (e-mail: eleguans@nus.edu.sg; elezfm@nus.edu.sg).

Digital Object Identifier 10.1109/TSMCB.2004.842247

in a robot-based navigation application, the robot needs to adapt to the new surroundings quickly and prompt actions may be required based on the new results.

In this paper, we propose an incremental approach to genetic algorithms, with different initialization schemes (ISs) based on different initialization of the GA population. We explore the use of incremental learning by implementing classifier agents in a multiagent environment, which means the classifier agents can collaborate to benefit each other. These approaches keep the old solutions and use an "integration" operation to integrate them with new elements to accommodate new attributes, while biased mutation and crossover operations are used to evolve further a reinforced solution. Four approaches with different initialization schemes are evaluated with several benchmark classification problems. The simulation results show that these approaches can be used successfully for incremental learning. They may also speed up the learning process and improve classification rates as compared to the retraining GA. Possible applications of proposed approaches for continuous incremental training and feature selection are also investigated.

Section II places our work in the context of related work. Section III discusses incremental learning in a multiagent environment and presents a model to illustrate the incremental approach. The basic GA for classification problems and various initialization schemes are elaborated in Section IV. The simulation results on benchmark databases and their analysis are illustrated in Section V. Discussions on proposed approaches are elaborated in Section VI. Section VII concludes the paper and presents future work.

II. Related Work

Many researchers have addressed incremental learning algorithms and methods. A wealth of work on incremental learning uses neural networks as learning subjects. Fu *et al.* [19] proposed an incremental backpropagation learning network that employs bounded weight modification and structural adaptation learning rules and applies initial knowledge to constrain the learning process. Yamauchi *et al.* [4] proposed incremental learning methods for retrieving interfered patterns. In their methods, a neural network learns new patterns with a relearning of a few number of retrieved past patterns that interfere with the new patterns. Polikar *et al.* [20] introduced $\mathrm{Learn}++$, an algorithm for incremental training of neural networks. Dalché-Buc and Ralaivola [21] presented a new local strategy to solve incremental learning tasks. It avoids relearning of all the parameters by selecting a working subset where the incremental learning is performed. Other incremental learning algorithms include the growing and pruning of classifier architectures [22] and the selection of most informative training samples [23].

GAs have been widely used in machine learning. DeJong *et al.* [7] considered the application of GAs to a symbolic learning task—supervised concept learning from a set of examples. Corcoran *et al.* [18] used GAs to evolve a set of classification rules with real-valued attributes. Ishibuchi *et al.* [9] examined the performance of a fuzzy genetics-based machine learning method for pattern classification problems with continuous attributes.

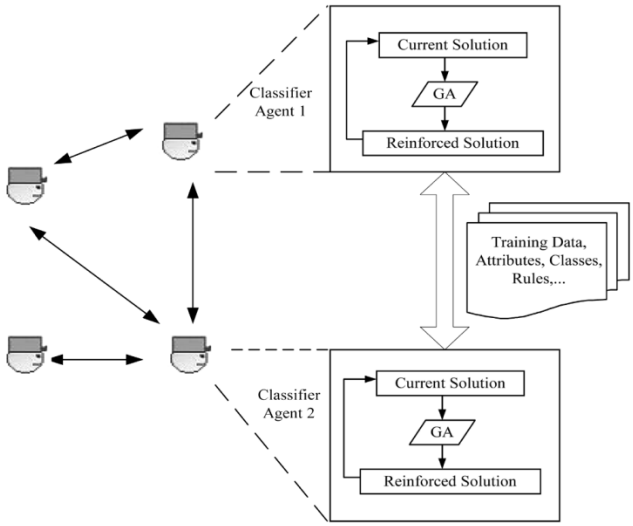


Fig. 1. Incremental learning of classifier agents with GA.

Their work only covers batch-mode algorithms, instead of incremental learning cases, while our work explores a method of using GAs with various initialization schemes in incremental learning.

There are two general approaches for GA-based rule optimization and learning. The *Michigan* approach uses GAs to evolve individual rules, a collection of which comprises the solution for the classification system [24]. Another approach is the *Pitt* approach [25], [26], where rule sets in a population compete against each other with respect to their performance on the domain task. The detailed explanation and comparison on these two approaches can be found in [27]. In this paper, the *Pitt* approach is chosen, as the encoding mechanism for this approach is more straightforward.

Another feature of this paper is that we place incremental learning in a context of multiagent environment, which means classifier agents may cooperate with each other to achieve incremental learning. There are two streams of research about combining multiagent systems and learning. One regards multiagent systems in which agents learn from the environment in which they operate. The second stream investigates the issues of multiagent learning with a focus on the interactions among the learning agents. Enee and Escazut [28] explored the evolution of multiagent systems with distributed elitism. It uses classifier systems as the evolution subjects. Caragea *et al.* [29] proposed a theoretical framework for the design of learning algorithms for knowledge acquisition from multiple distributed, dynamic data sources.

III. Incremental Learning With GAs

In some classification problems, new training data, attributes and classes may become available or some existing elements may get changed. Thus, classifier agents should have algorithms to cope with these changes. Either they may sense the environment and evolve their solutions by themselves, or they may collaborate to adapt to the new environment, as shown in Fig. 1. There are many possible types of cooperation among a group of agents to boost their capability. Classifier agents can

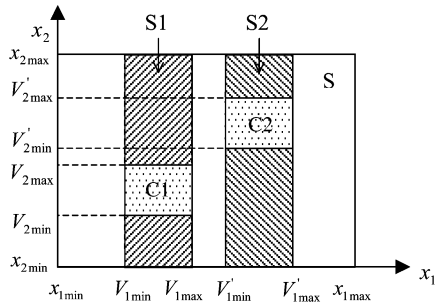


Fig. 2. ILGA model for a simplified classification problem.

exchange information of new attributes and classes. If available, they can also exchange evolved rule sets (chromosomes). They can even provide each other with new training/test data, or challenge other agents with unsolved problems. Various combinations of these operational modes are also possible.

Fig. 1 also shows the use of the GA as the main approach for incremental learning, either with self-learning or collaborative learning. Each classifier agent may have a certain solution (current solution in the figure) based on the attributes, classes, and training data currently known. When new attributes, classes, or data are sensed or acquired from the environment or other agents, the GA is then used to learn the new changes and evolve into a reinforced solution. As long as the learning process continues, this procedure can be repeated for incremental learning.

We postulate that incremental learning with genetic algorithms (ILGA) can improve classification rate and save training time. The following gives some insight and explanation on the inner mechanism leading to the advantages of ILGA.

Fig. 2 presents a model for a simplified classification problem with two attributes (x_1, x_2) and two classes (C_1, C_2). This simplified model and the following analysis can be easily extended to higher-dimensional spaces. S denotes the possible area confined by the value range of $(x_{1\min}, x_{1\max})$ and $(x_{2\min}, x_{2\max})$, which is also the maximum searching area for the GA. C_1 and C_2 are the areas covering the training instances belonging to the two class categories respectively. When the normal GA evolves the rule set using two attributes (x_1 and x_2) together, it searches the area S directly. However, ILGA consists of two steps. The first step is a one-dimensional search along the axis x_1, trying to find the boundary information for both classes, i.e., $V_{1\min}$, $V_{1\max}$, $V'_{1\min}$, $V'_{1\max}$. The second step inherits these boundary information, and continue searching the boundary information for the two attributes (further searching $V_{2\min}$, $V_{2\max}$, $V'_{2\min}$, $V'_{2\max}$). However, with the help of the information inherited, the second search step can be confined in the areas $S1$ and $S2$ (the shadow area plus the grid area for each as shown in the figure). It is easy to see that the search space is largely reduced from S to $S1 \bigcup S2$. Despite the overhead of the one-dimensional search in the first step, ILGA still searches a smaller area than the normal GA. This explains why ILGA needs less training time. That is, as ILGA inherits useful information; it stands on a better starting point.

Fig. 2 also provides some insight on the possibility of improvement on the classification rates. As the initial population for ILGA is created using the boundary information of x_1, they are located in the area $S1$ or $S2$. Because they are closer to the best solutions C_1 or C_2, it is more likely that ILGA converges to either solution. However, the normal GA needs to search a larger area S, thus it may miss the best solutions occasionally. As shown in Fig. 2, it is relatively more difficult for the normal GA to derive such solution contour. As a result, the partitioning in the attribute domain brings along some advantages. In general, the interference among interfering attributes makes the GA search more difficult. When a larger attribute domain is partitioned, the interference among attributes can be reduced. Therefore, it is easier to map partial attributes to classes, which makes ILGA-based search easier and more accurate.

Furthermore, from Fig. 2, ILGA should stick to the $S1$ neighborhood for the old attribute x_1, while exploring more for the boundary information on the new attribute x_2. If the exploration is more focused on the new attribute, the training time can be shortened. This motives us to reduce the mutation and crossover rates. That is, if the point chosen for mutation or crossover is located in the old attribute portion, the corresponding rate will be reduced. The detailed design will be presented in Section IV-E.

The advantages of ILGA and the motivation for reduced rates can also be explained with the schema analysis and building block hypothesis [16], [30]. A schema is a similarity template describing a subset of strings with similarities at certain string positions. It is postulated that an individual's high fitness is due to the fact that it contains good schemata. Short and high-performance schemata are combined to form building blocks with higher performance expected. Building blocks are propagated from generation to generation, which leads to a keystone of the GA approach. Research on the GA has proved that it is beneficial to preserve building blocks during the evolution process [30]. ILGA inherits the old chromosomes from the previous results, where the building blocks likely reside. The integration of these building blocks into the initial population provides a solid foundation for the following evolutions. Also, the smooth preservation of these building blocks during the following evolutions also boosts the classification performance. This justifies the use of the reduced rates. When the crossover and mutation rates are reduced in the old elements portion, the building blocks inside will undergo less genetic evolution pressure and thus increase their survival chance.

IV. Design of Incremental Learning With GAs

Our approaches are developed on the basis of standard GAs. The design of GAs for rule-based classification is presented first, then different initialization schemes for incremental learning are elaborated. A typical GA is shown in Fig. 3. The genetic operators including mutation, crossover, and selection are presented in Appendix I, which provides a detailed explanation of mutationRate, crossoverRate, and survivorPercent.

The task of classification is to assign instances to one out of a set of predefined classes, by discovering certain relationship among attributes. Let us assume our pattern classification problem is a c-class problem in an n-dimensional pattern space. And q real vectors $X_i = (x_{i1}, x_{i2}, \ldots, x_{in})$, $i = 1, 2, \ldots, q$, are given as training patterns from the c classes $(c \ll q)$. Normally, a learning algorithm is applied to a set of training data

```
begin
t:=0;
initialize P(t);   //initialise a population of candidates
evaluate P(t);   //evaluate each candidate using a fitness function
while (not terminate-condition) do      //stopping criteria
      begin
         select P'(t) from P(t);
         crossover P'(t);
         mutate P'(t);
         combine P'(t) and P(t) to form P(t+1);
         evaluate P(t+1);
         t:=t+1;
      end
end
```

Fig. 3. Pseudocode of a typical GA.

with known classes to discover the relationship between the attributes and classes. The discovered rules can be evaluated by classification accuracy or error rate either on the training data or test data.

For classification problems, the discovered rules are usually represented in the following IF-THEN form:

$$\texttt{IF}\langle\texttt{condition 1}\rangle\&\langle\texttt{condition 2}\rangle\&\ldots \&\langle\texttt{condition n}\rangle\ \texttt{THEN}\ \langle\texttt{action}\rangle. \quad (1)$$

Each rule has one or more conditions as the antecedent, and an action statement as the consequent which determines the class category. There are various representation methods for the conditions and actions in terms of the rule properties (fuzzy or nonfuzzy) and the attribute properties (nominal or continuous). In this paper, we use nonfuzzy IF-THEN rules. A rule set which consists of a certain number of rules is supposed to be a solution for a classification problem.

A. Encoding Mechanism

In our approach, an IF-THEN rule is represented as follows:

$$R_i: \text{IF } (V_{1\min} \le x_1 \le V_{1\max}) \wedge (V_{2\min} \le x_2 \le V_{2\max}) \cdots \wedge (V_{n\min} \le x_n \le V_{n\max}) \text{ THEN } y = C \quad (2)$$

where R_i is a rule label, n is the number of attributes, $(x_1, x_2, \ldots x_n)$ is the attribute set, and C is a class. $V_{j\min}$ and $V_{j\max}$ are the minimum and maximum bounds of the jth attribute x_j respectively. We encode rule R_i according to the equation shown at the bottom of the page, where Act_j denotes whether the condition j is active or inactive, which is encoded as 1 or 0. If $V_{j\min}$ is larger than $V_{j\max}$ at any time, this element will be regarded as an invalid element.

Each antecedent element represents an attribute, and each consequence element stands for a class. Each chromosome CR_j consists of a set of classification rules R_i $(i = 1, 2 \ldots, m)$ by concatenation

$$CR_j = \bigcup_{i=1,m} R_i \quad j = 1, 2, \ldots, p \quad (3)$$

where m is the maximum number of rules allowed for each chromosome and p is the size of the population. Therefore, one chromosome will represent one rule set. Assume we know the value range for each attribute and class *a priori*, $V_{j\min}$, $V_{j\max}$, and C can be encoded each with a character by finding their positions in the ranges. Thus, the final chromosome can be encoded as a string consisting of characters. According to the above encoding mechanism, each chromosome will consist of L characters, where

$$L = m*(3*n + 1). \quad (4)$$

If all the antecedent elements in a rule are inactive, this rule will be regarded as a noncontributing rule. With this mechanism, our approach has the feature of variable-length GAs so that the number of active rules in a rule set can be flexible. This encoding mechanism is suitable for classification problems whose attributes are real valued. However, it can be easily extended to classification problems with nominal-valued attributes.

B. Fitness Function

The fitness of a chromosome reflects the success rate achieved while the corresponding rule set is used for classification. The GA operators use this information to evolve better chromosomes over generations. As each chromosome in our approach comprises an entire rule set, the fitness function actually measures the collective behavior of the rule set. The fitness function simply measures the percentage of instances that can be correctly classified by the chromosome's rule set.

Since there is more than one rule in a chromosome, it is possible that multiple rules match the conditions for all attributes but predict different classes. We use a voting mechanism to resolve conflict. That is, each rule casts a vote for the class predicted by itself, and finally the class with the highest votes is regarded as the conclusive result. If any classes tie on one instance, it is then concluded that this instance cannot be classified correctly by this rule set. (Our observation is that this case rarely happens, therefore it will not hurt the accuracy performance much.) Fig. 4 shows the pseudocode for fitness evaluation.

C. Stopping Criteria

There are four factors in the stopping criteria. The evolution process stops after a generation limit, or when the best fitness of chromosome reaches a preset threshold (which is set as 1.0 through this paper), or when the best fitness of chromosome has

Antecedent Element 1			.....	Antecedent Element n			Consequence Element
Act_1	$V_{1\min}$	$V_{1\max}$	.....	Act_n	$V_{n\min}$	$V_{n\max}$	C

```
correctNumber:=0;
        for each instance
        { for each rule in the chromosome
                {decode rule antecedents;
                if (all rule antecedents are valid for the instance)
                  then cast a vote for the class reported in the rule ;
                }
          use a voting mechanism to determine the classPredicted;
        if (classPredicted class==realClass in the instance)
         then  correctNumber++;
        }
fitness:=correctNumber/totalNumber;   //totalNumber is the total number of instances
```

Fig. 4. Pseudocode for evaluating the fitness of one chromosome.

```
Loop begin
if new attributes are ready to be integrated
   t:=0;
   if group chromosomes in current solution are available
       then select them as seeds;
            else select the chromosome with best fitness as the seed;
   if new chromosomes for new attributes are available to be integrated
       then integrate them with the old chromosomes;
            else expand old chromosomes with some randomly created elements;
   evaluate P(t); // P(t) is the resulting population from the  above steps
   while (not terminate-condition) do        //stopping criteria
         begin
            select P'(t) from P(t);
            biased crossover P'(t);   //  biased crossover rate used
            biased mutation P'(t);    // biased mutation rate used
            combine P'(t) and P(t) to form P(t+1); // based on the survivorsPercent rate
            evaluate P(t+1);
            t:=t+1;
         end
Loop end
```

Fig. 5. Incremental approach to the GA with initialization schemes.

no improvement over a specified number of generations—stagnation limit, or the current performance on the validation data has degraded for 10% compared to the average performance of the previous 20generations, if the validation data are used. The detailed settings are reported along with the corresponding results.

D. Initial Population With Different ISs

Fig. 5 shows the pseudocode of our incremental approach to the GA with initialization schemes. The main features lie in the formation of the initial population, integration of old and new chromosomes, the biased genetic operators, and incremental evolution of new attributes. Group chromosomes in the figure refer to the chromosomes preserved in classifier agents as the result from the last round of evolution.

The formation of initial population is one of the main features of our approaches, in which the integration of old and new chromosomes/elements is the major contribution. Fig. 6 shows how the new elements are inserted into an old rule to form a new rule. Note that it only shows the operation on a single rule for the purpose of simplicity. The other rules in the chromosome will undergo similar operations.

There are several ways to construct new chromosome population in terms of the selection of old chromosome(s) and newly appended elements. For the old chromosome(s), we can either use the best rule set (chromosome) as a seed for all the initial population or the whole population of chromosomes in the current solution if available. To create new elements, we have two choices. We can append randomly created new elements to the old rule sets, if the new information acquired or exchanged from the other classifier agent only includes the new attributes and

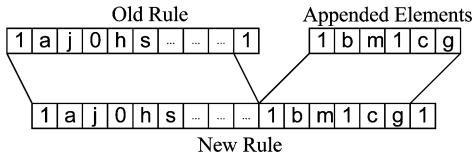
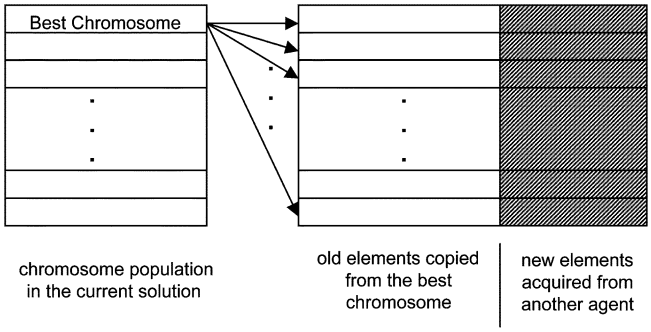


Fig. 6. Formation of a new rule in a chromosome.



(a)

```
for each newChrom[j] in the new population
 { bufferChrom=the best chromosome from the current solution;
   incomingChrom:=a  chromosome randomly selected from the group
                     chromosomes coming from another agent;
   for each rule i in bufferChrom
          { curClass:=the class of rule i;
            analyze incomingChrom, and  place all new incoming rules having the
                    same class as curClass into a candidate pool;
            randomly select a rule from the candidate  pool;
            insert all  the elements for the new attributes in the selected rule into
                    bufferChrom;
          }
   newChrom[j]:=bufferChrom;
 }
```

(b)

Fig. 7. (a) Illustration for integrating old chromosomes with new elements under IS2. (b) Pseudocode for integrating old chromosomes with new elements under IS2.

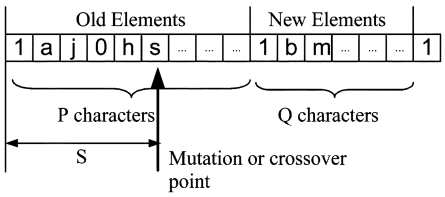
their value ranges. If the other classifier agent can provide more information such as the entire evolved rule set covering the new attributes, it will be more helpful to use the elements from such a rule set. We list these choices in Table I, and give them distinct names for comparison later (IS in the table stands for initialization schemes). Actually, there are many options for the integration of old and new elements. For example, the elements from the acquired rules can also be available as the best chromosome or group chromosomes. Further discussion on this is covered in Section VI.

Fig. 7(a) illustrates the formation of a new population using IS2 and Fig. 7(b) shows the corresponding pseudocode of how the new chromosomes are created by integrating the old and new elements under IS2. We can see that IS2 copies from the current solution the best chromosome into all the new chromosomes, and new elements are selected with a matching mechanism from the incoming chromosomes from the other agent. The pseudocodes for IS1, IS3, and IS4 are listed in Appendix II.

These four alternatives are applicable to different environments for incremental learning. If incremental learning happens in a stand-alone agent, this agent can only use IS1 and IS3. However, in a collaborative multiagent environment, classifier agents gain more freedom on choosing which approach to use. They

TABLE I
ALTERNATIVES ON THE FORMATION OF A NEW POPULATION

Old Elements \ New Elements	Randomly Created Elements	Elements from Acquired Rules
Best Chromosome	IS1	IS2
Group Chromosomes	IS3	IS4



$$P_m^* = \begin{cases} \alpha \cdot P_m & 1 \le S \le P \\ P_m & P < S \le P+Q \end{cases} \qquad P_C^* = \begin{cases} \beta \cdot P_C & 1 \le S \le P \\ P_C & P < S \le P+Q \end{cases}$$

$$0 \le \alpha, \beta \le 1$$

Legends: S - mutation or crossover point selected;
P - length of the old elements; Q - length of the new elements;
α - mutation reduction rate; β - crossover reduction rate;
P_m - mutation rate applied to the new elements (mutationRate);
P_c - crossover rate applied to the new elements (crossoverRate);
P_m^* - biased mutation rate applied to the old elements;
P_c^* - biased crossover rate applied to the old elements.

Fig. 8. Biased crossover and mutation rates.

may choose one of the four approaches according to the environmental situation. We will have a detailed discussion on this later based on the experiment results.

E. Biased Mutation and Crossover

In our approaches, we have chosen to bias the mutation and crossover operators with some preference toward the new elements. Mutation and crossover points are still selected randomly. However, if the point chosen for mutation or crossover is located in the old elements part, the corresponding rates may be reduced with a reduction rate, as shown in Fig. 8. The mutation and crossover reduction rates are called α and β, respectively.

The motivation behind this is that we tend to preserve the structure of old elements and explore more on the combination between old and new elements. The old elements part still needs variation, but the rates applied can be comparatively lower, compared with the rates exerted on the new elements. Classifier agents can adjust the values for α and β for different classification problems to achieve better solutions. We have more experiments and discussions on the setting of α and β in Section V-B.

V. EXPERIMENTAL RESULTS AND POSSIBLE APPLICATIONS

A. Experiment Scheme

Ten benchmark data sets have been used for experiments in this paper. The information of these data sets is provided in Table II, which includes the number of instance, attributes, and classes in each data set. The first nine sets are taken from the UCI machine learning repository [31], and the last one is taken from the PROBEN1 collection [32]. They all are real-world problems.

TABLE II
DATASETS USED FOR THE EXPERIMENTS

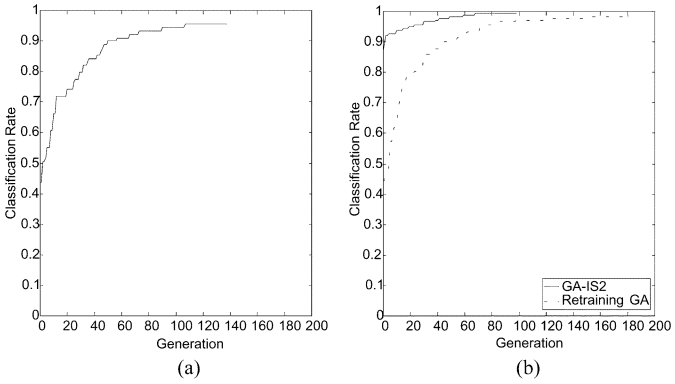
Data Set	No. of Instances	No. of Attributes	No. of Classes
Wine	178	13	3
Glass	214	9	6
Cancer	699	9	2
Yeast	1484	8	10
Boston Housing	506	13	3
Waveform	5000	21	3
Image Segmentation	2310	19	7
Vehicle	846	18	4
Bupa	345	6	2
Diabetes	768	8	2

All experiments were completed on Pentium III 650-MHz PCs. The mean values reported were averaged over ten independent runs. Standard deviation (SD) is reported in braces. The t-test is used to evaluate the significance of difference between two means, and t-value, p-value, and degree of freedom (df) are reported for a certain claim. $p = 0.05$ is used as the threshold for significance of difference, i.e., the difference is significant with at least 95% confidence level, if its p-value with t-test is less than 0.05.

B. Results and Analysis

In this section, we evaluate the performance of ILGA. The data sets used in this section were partitioned into four parts, namely, TRA, TRB, VAL, and TST, each with 25% of instances. TRA and TRB are used as training data, VAL is used as validation data, and TST is used as unknown test data. Therefore, when the classifier knows only some partial attributes, it uses TRA, VAL, and TST with those partial attributes. When it learns the full set of attributes, TRA + TRB with the complete attributes are then used as training data, VAL and TST with the complete attributes are used as validation and test data.

Fig. 9(a) shows the evolution of the best rule set with the first ten attributes in the wine data set. The figure records the best CR in each generation, i.e., the highest fitness value achieved for each generation. It is shown that CR grows from an initial value of 0.44 gradually, and finally reaches 0.95 at generation 140. Fig. 9(b) shows the succeeding GA process with 13 attributes. IS2 is chosen for the formation of the initial population, which uses the best chromosome in the resultant rule set from (a), and combining it with the rule elements from the other agent. The retraining GA, which trains the classifier from scratch (i.e., with



Notes:
1. for (a), mutationRate=0.01, crossoverRate=1, survivorsPercent=50%, ruleNumber=30, popSize=200, stagnationLimit=30, generationLimit=200.
2. for (b), α =0.5, β =0.5, the rest are the same as in Note 1.

Fig. 9. Simulation results show: (a) classifier agent evolving rule sets with ten attributes and (b) GA with IS2 running to achieve rule sets with 13 attributes, compared to the retraining GA approach.

random initialization) again with 13 attributes, is also shown in the figure for comparison.

We can note from the figure that the best CR from the GA with IS2 decreases from 0.95 to 0.87 immediately after the formation of initial population. This can be explained by the facts that new attributes are integrated into the new chromosomes and the new training data are used. Then, CR increases gradually, and reaches 0.99 with 100 generations. In the case of retraining GA, it costs about 180 generations to reach a CR of 0.98. As a result, we find that the GA with IS2 has integrated successfully the new attributes, and evolves a new rule set within shorter time compared to the retraining GA.

Table III compares performance between the four IS approaches and the retraining GA using the wine data. The GA was run initially with a partial number of attributes first (nine and ten attributes as shown in the table). NOA and NNA represent a different combination of the number of old and new attributes. The attributes were used in the same order as in the original data set. For instance, $\mathrm{NOA} = 10$ means the first ten attributes were used as old attributes, and $\mathrm{NNA} = 3$ means the remaining three attributes were used as new attributes. With the results of the GA, four IS approaches and the retraining GA were conducted. Therefore these approaches have the same starting point, which is fair for comparison.

We find that all four IS approaches perform well in integrating various numbers of attributes. For example, in the case of $\mathrm{NOA} = 9$, $\mathrm{NNA} = 4$, the initial run of GA achieves an ending CR of 0.9477 and a test CR with 0.6511. IS4 gets an initial CR of 0.8011. Then with about 70 generations, it reaches an ending CR of 0.9864 and a test CR of 0.8222. This means that IS4 recovers the information loss caused by missing attributes, and obtains new capability to use the information from the expanded attributes.

With a comparison among the four IS approaches and the retraining GA, it is found that all the four IS approaches cost fewer generations (shorter training time) than the retraining GA, which demonstrates the advantages of GAs with specialized initialization schemes. T-tests on training time show the significance of these advantages, e.g., $t = 3.93$, $p < 0.001$, $\mathrm{df} = 18$, for IS4 versus the retraining GA in $\mathrm{NOA} = 9$; $t = 3.34$, $p < 0.01$, $\mathrm{df} = 18$, for IS2 versus the retraining GA in $\mathrm{NOA} = 9$; $t = 3.93$, $p < 0.001$, $\mathrm{df} = 18$, for IS4 versus the retraining GA in $\mathrm{NOA} = 10$; $t = 3.76$, $p < 0.01$, $\mathrm{df} = 18$, for IS2 versus the retraining GA in $\mathrm{NOA} = 10$.

TABLE III
COMPARISON OF THE PERFORMANCE OF THE GA WITH DIFFERENT IS APPROACHES ON THE WINE DATA

NOA=9, NNA=4	Partial Attributes	Initial CR	0.4614 (0.0402)				
		Generations	75.9 (13.9)				
		T. time (s)	105.1 (20.9)				
		Train. CR	0.9477 (0.0264)				
		Test CR	0.6511 (0.1109)				
			Retraining GA	IS1	IS2	IS3	IS4
	Full Attributes	Initial CR	0.3989 (0.0270)	0.7 (0.0458)	0.7943 (0.0468)	0.6852 (0.0491)	0.8011 (0.0483)
		Generations	115.4 (29.1)	102.5 (26.3)	75.2 (21.4)	85.1 (14.8)	71.0 (18.1)
		T. time (s)	292.2 (78.0)	258.4 (64.5)	192.6 (53.1)	211.8 (35.7)	179.8 (45.5)
		Train. CR	0.9784 (0.0230)	0.9716 (0.0154)	0.9852 (0.0108)	0.975 (0.0159)	0.9864 (0.0117)
		Test CR	0.7489 (0.0618)	0.7522 (0.0792)	0.7911 (0.0846)	0.8022 (0.0851)	0.8222 (0.0679)
NOA=10, NNA=3	Partial Attributes	Initial CR	0.4500 (0.0477)				
		Generations	82.1 (20.1)				
		T. time (s)	119.2 (29.7)				
		Train. CR	0.9477 (0.0877)				
		Test CR	0.6800 (0.1011)				
			Retraining GA	IS1	IS2	IS3	IS4
	Full Attributes	Initial CR	0.3989 (0.0270)	0.7341 (0.0458)	0.7819 (0.0638)	0.7545 (0.0611)	0.7886 (0.0580)
		Generations	115.4 (29.1)	90.6 (25.3)	73.8 (20.5)	89.4 (15.0)	70.0 (20.6)
		T. time (s)	292.2 (78.0)	229.3 (68.0)	180.4 (52.5)	223.5 (36.2)	177.4 (49.4)
		Train. CR	0.9784 (0.0230)	0.9761 (0.0136)	0.9852 (0.0152)	0.9489 (0.0793)	0.9818 (0.0187)
		Test CR	0.7489 (0.0618)	0.7667 (0.0764)	0.8133 (0.0908)	0.8089 (0.0639)	0.8256 (0.0627)

NOA: number of old attributes; NNA: number of new attributes;

Notes:
1. The meaning of various IS approaches can be referred to Table I.
2. mutationRate=0.01, crossoverRate=1, survivorsPercent=50%, ruleNumber=30, popSize=200, stagnationLimit=30, generationLimit=200, α =0.5, β =0.5.
3. "Initial CR" means the best classification rate achieved by the initial population on the training data;
"Generations" means the number of generations needed to reach stopping criteria;
"T. time (s)" means the training time cost, in seconds;
"Train. CR" means the best classification rate achieved by the resulting population on the training data;
"Test CR" means the classification rate achieved by the resulting population on the test data.

It is found from Table III that the mean of test CRs of the four IS approaches are larger than that of the retraining GA. We also find the standard deviations of test CR become larger, compared to the training CR. This is reasonable as the test data are totally unknown to the classifier agents during the training process. As a result, some t-tests on the test CR shows that the difference between the IS approaches and the retraining GA is not significant. For example, $t = 1.86$, $p < 0.1$, $\mathrm{df} = 18$ for IS2 versus the retraining GA in $\mathrm{NOA} = 10$. But a few t-tests still show the significance of difference. For instance, IS4 achieves better test CR than the retraining GA ($t = 2.52$, $p < 0.05$, $\mathrm{df} = 18$, for $\mathrm{NOA} = 9$; $t = 2.76$, $p < 0.05$, $\mathrm{df} = 18$, for $\mathrm{NOA} = 10$).

Table III also shows the comparison among the four IS approaches (as listed in Table I). The results show that IS2 and IS4 cost fewer generations and thus less training time to reach convergence than IS1 and IS3. For instance, IS4 uses less training time than IS1 ($t = 3.33$, $p < 0.01$, $\mathrm{df} = 18$, for $\mathrm{NOA} = 10$; $t = 3.14$, $p < 0.01$, $\mathrm{df} = 18$ for $\mathrm{NOA} = 9$). This may be explained by recalling the method each approach uses to form the initial population. As IS2 and IS4 use the entire evolved rule set from the other agent, they can acquire more useful information from the rule set than IS1 and IS3 with the randomly created elements. This can also be verified by observing the initial CR for

TABLE IV
COMPARISON OF THE PERFORMANCE OF THE GA WITH DIFFERENT IS APPROACHES ON THE CANCER DATA

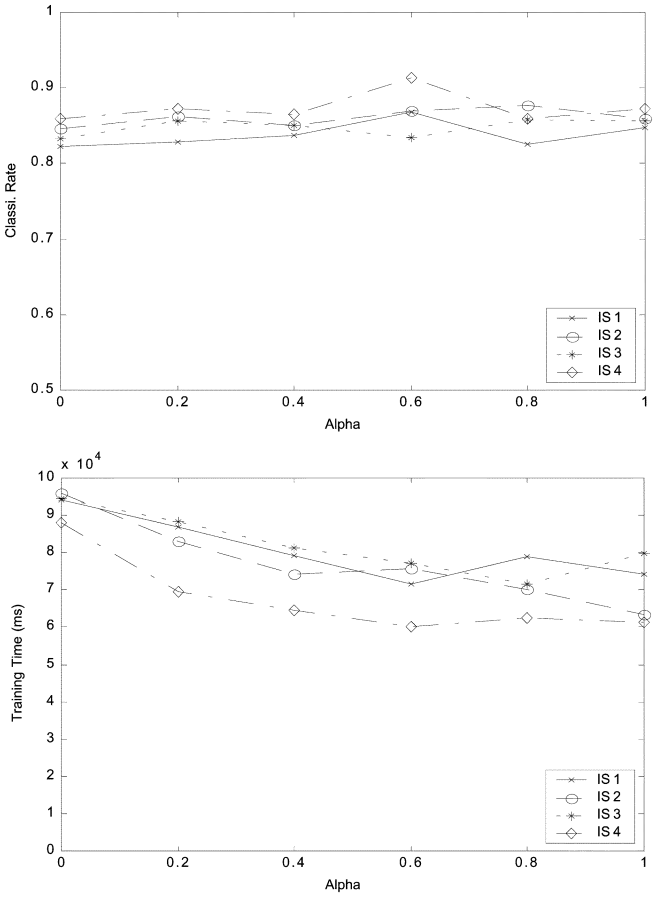
NOA=6, NNA=3	Partial Attributes	Initial CR	0.8389 (0.0402)				
		Generations	57.4 (9.3)				
		T. time (s)	165.2 (30.6)				
		Train. CR	0.9766 (0.0032)				
		Test CR	0.9594 (0.0130)				
			Retraining GA	IS1	IS2	IS3	IS4
	Full Attributes	Initial CR	0.7840 (0.0343)	0.9189 (0.0121)	0.9350 (0.0115)	0.9298 (0.0124)	0.9364 (0.0103)
		Generations	104.0 (18.5)	92.8 (16.1)	65.8 (13.0)	83.1 (15.8)	63.0 (17.0)
		T. time (s)	450.0 (85.3)	391.6 (68.7)	272.1 (55.5)	349.6 (78.7)	275.9 (82.2)
		Train. CR	0.9679 (0.0026)	0.9679 (0.0071)	0.9673 (0.0041)	0.9688 (0.0042)	0.9679 (0.0035)
		Test CR	0.9629 (0.0098)	0.9629 (0.0159)	0.9674 (0.0179)	0.9623 (0.0158)	0.9691 (0.0138)
NOA=4, NNA=5	Partial Attributes	Initial CR	0.8549 (0.0380)				
		Generations	59.0 (9.7)				
		T. time (s)	105.6 (21.5)				
		Train. CR	0.9726 (0.0024)				
		Test CR	0.9440 (0.0100)				
			Retraining GA	IS1	IS2	IS3	IS4
	Full Attributes	Initial CR	0.7840 (0.0343)	0.8748 (0.0192)	0.9350 (0.0079)	0.8871 (0.0226)	0.9304 (0.0120)
		Generations	104.0 (18.5)	93.2 (20.3)	69.6 (20.5)	83.6 (17.1)	60.1 (13.3)
		T. time (s)	450.0 (85.3)	393.3 (86.6)	287.8 (87.5)	351.7 (85.2)	263.2 (64.3)
		Train. CR	0.9679 (0.0026)	0.9673 (0.0056)	0.9685 (0.0038)	0.9670 (0.0020)	0.9662 (0.0030)
		Test CR	0.9629 (0.0098)	0.9663 (0.0078)	0.9663 (0.0133)	0.9657 (0.0134)	0.9720 (0.0074)

Notes:
1. mutationRate=0.01, crossoverRate=1, survivorsPercent=50%, ruleNumber=30, popSize=200, stagnationLimit=30, generationLimit=200, α =0.5, β =0.5.
2. Other legends follow the explanations under Table III.

the IS approaches, i.e., the initial CRs of IS2 and IS4 are higher than those of IS1 and IS3.

Table IV compares the performance of various IS approaches on another data set—the cancer data. We still get similar findings as those obtained from Table III. We find that the four IS approaches meet the requirements of incremental learning with different performance. The comparison between the IS approaches and retraining GA also shows similar findings as observed from the wine data. For instance, IS4 outperforms the retraining GA in training time ($t = 4.64$, $p < 0.001$, $\mathrm{df} = 18$, for $\mathrm{NOA} = 6$; $t = 5.53$, $p < 0.001$, $\mathrm{df} = 18$, for $\mathrm{NOA} = 4$), and IS2 outperforms the retraining GA in training time ($t = 5.52$, $p < 0.001$, $\mathrm{df} = 18$, for $\mathrm{NOA} = 6$; $t = 4.20$, $p < 0.001$, $\mathrm{df} = 18$, for $\mathrm{NOA} = 4$). IS2 and IS4 again need fewer generations (less training time) to reach convergence. For instance, IS2 uses less training time than IS1 ($t = 4.28$, $p < 0.001$, $\mathrm{df} = 18$, for $\mathrm{NOA} = 6$; $t = 2.71$, $p < 0.02$, $\mathrm{df} = 18$, for $\mathrm{NOA} = 4$). T-tests on the test CRs also reported different significance levels on the performance difference between the IS approaches and the retraining GA. For instance, the advantage of IS4 over the retraining GA is significant ($t = 2.33$, $p < 0.05$, $\mathrm{df} = 18$, for $\mathrm{NOA} = 4$), while the advantage of IS2 over the retraining GA is not significant ($t = 0.67$, $p > 0.2$, $\mathrm{df} = 18$, for $\mathrm{NOA} = 4$). We find the advantages of IS approaches here are mainly reduced training time.

Finally, we conducted several experiments with different settings of the mutation and crossover reduction rates α and β to explore their effects on the performance of various IS approaches. When one rate is evaluated, another rate is fixed as 1.0. Figs. 10 and 11 show the results on the wine data with different values of α and β respectively. $\alpha = 0$ or $\beta = 0$ means there is no operation (mutation or crossover) on the old elements, and $\alpha = 1$ or $\beta = 1$ means there is no bias on mutation or crossover between the old elements and new elements. We find that α and β really affect the performance of IS approaches. Both figures show that if IS approaches are used with $\alpha = 0$ or $\beta = 0$, they



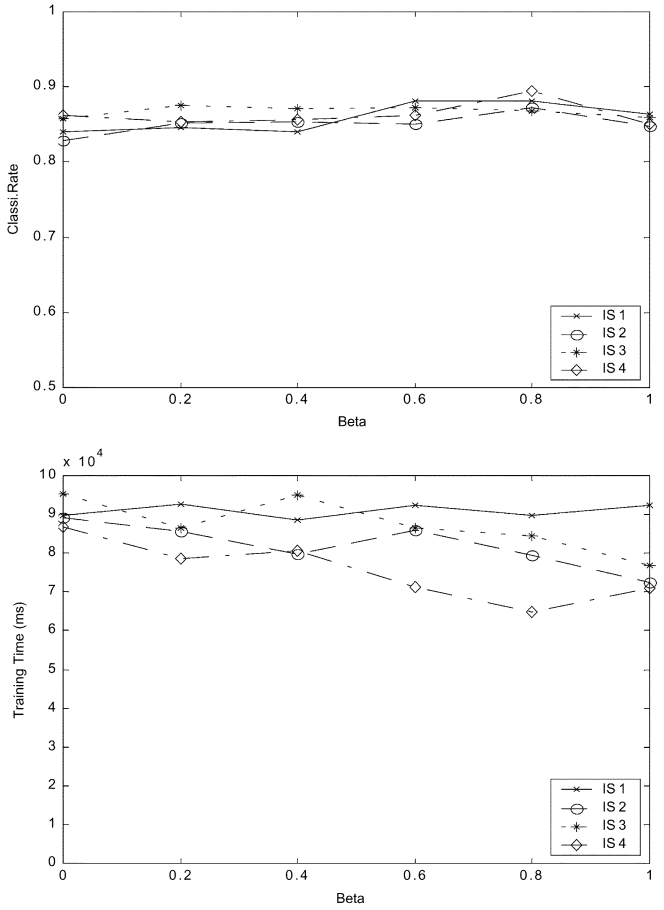
Notes:
1. mutationRate=0.01, crossoverRate=1, survivorsPercent=50%, ruleNumber=30, popSize=200, stagnationLimit=30, generationLimit=150, NOA=10, NNA=3.

Fig. 10. Effect of mutation reduction rate α on the performance of IS approaches (test CR and training time) with the wine data.

need the longest training time and achieve lower test CRs compared to other values for α and β. This tells us that the extremely biased rates ($\alpha = 0$ or $\beta = 0$) are not suitable for IS approaches and the old elements still need some genetic operations. It is also shown in both figures that and training time decreases in a general trend, when the values of α and β increase. But the improvement on the test CR is very small and can be neglected. The best values for α and β are between 0.6 to 0.8, depending on the type of IS approaches used. This result supports the use of reduced crossover and mutation rates on the old elements.

Recalling the analysis in Section III based on Fig. 2, we have explained the motivation and necessity of two reduction rates, i.e., $\alpha < 1$ and $\beta < 1$. Meanwhile, as the real classification problems are more complicated than the simplified one, the exploration on the boundary information of the old attribute should not be stopped entirely. This implies $\alpha > 0$ and $\beta > 0$. Thus, we have $0 < \alpha < 1$ and $0 < \beta < 1$, and the optimal values for the reduction rates exist in between. The above experiment results have confirmed this analysis. With the selection of α and β, the generation cost to achieve the stopping criteria can be reduced, thus the training time will be saved accordingly.

We have evaluated the performance of ILGA with other six data sets. Due to the limited space, we just compare the performance between IS4 and retraining GA, and only one attribute



Notes:
1. mutationRate=0.01, crossoverRate=1, survivorsPercent=50%, ruleNumber=30, popSize=200, stagnationLimit=30, generationLimit=150, NOA=10, NNA=3.

Fig. 11. Effect of crossover reduction rate β on the performance of IS approaches (test CR and training time) with the wine data.

partition is tried for each data set. (If the total number of attributes is an even number $(2n)$, half attributes are simulated as old attributes, and the other half as new attributes; if it is an odd number $(2m+1)$, then $m+1$ attributes are used as old attributes, while the rest as new ones.) The experimental results are shown in Table V, which confirm again the above findings. That is, IS4 outperforms the retraining GA in training and test CRs, and it also costs less training time with most data sets. The improvement on the classification performance is larger for some data sets such as the vehicle and the image segmentation.

C. Further Applications of Incremental Learning With the GA

The above experiments have shown that the incremental learning with GAs is feasible with the specially-designed algorithms. Four different approaches have been proposed and compared. In this section, we explore further applications in two aspects, namely, continuous incremental training and feature selection.

In the earlier experiments, it is assumed that only one set of new attributes is introduced, and they are treated as a batch. Actually these algorithms can be extended to run in the continuous mode, which means new attributes can be introduced one after another. The resulting classification rule sets are also evolved incrementally to accommodate the new attributes. The new approaches are termed as continuous incremental genetic algorithms (CIGAs). With the incorporation of the four IS approaches respectively, four approaches of CIGAs from CIGA1 to CIGA4 are available.

TABLE V
ILGA RESULTS ON SOME DATA SETS

Data Set		Re-training GA	IS4
Bupa	T. time (s)	373.5 (167.5)	371.6 (170.3)
	Train. CR	0.6558 (0.0352)	0.7302 (0.0503)
	Test CR	0.6034 (0.0267)	0.6287 (0.0287)
Vehicle	T. time (s)	1351.7 (66.8)	1300.7 (170.3)
	Train. CR	0.5436 (0.0613)	0.6421 (0.0402)
	Test CR	0.5079 (0.0562)	0.5944 (0.0308)
Waveform	T. time (s)	7342.9 (311.1)	6287.5 (1352.4)
	Train. CR	0.6828 (0.0625)	0.7638 (0.0633)
	Test CR	0.6591 (0.0656)	0.7327 (0.0584)
Boston Housing	T. time (s)	224.4 (58.8)	215.3 (32.0)
	Train. CR	0.6944 (0.0876)	0.8206 (0.0275)
	Test CR	0.4102 (0.0544)	0.4654 (0.0325)
Image Segmentation	T. time (s)	847.2 (414.4)	772.5 (417.1)
	Train. CR	0.5863 (0.1701)	0.6832 (0.1214)
	Test CR	0.5725 (0.1630)	0.6777 (0.1186)
Yeast	T. time (s)	1255.2 (470.0)	1104.0 (439.1)
	Train. CR	0.3984 (0.0484)	0.4785 (0.0356)
	Test CR	0.3588 (0.0361)	0.4067 (0.0228)

Fig. 12 compares the performance of four types of CIGAs on the glass data in terms of training CR and test CR respectively. It is shown that all types of CIGAs outperform the normal GA in both CRs. Among the four types, CIGA4 is the best approach, and CIGA1 and CIGA3 are inferior to CIGA2 and CIGA4 in both CRs. The simulation results showed that CIGAs can be used successfully for continuous incremental training of classifier agents and can achieve better performance than the normal GA using batch-mode training.

In the experiments mentioned, the whole attribute set is divided into old and new parts. However, the sequence of attributes is kept unchanged as in the original data sets. In fact, the sequence of the attributes does affect the final performance. We have conducted thorough investigation on the impact of attribute ordering in [33]. In the new approach, called ordered incremental genetic algorithm (OIGA), attributes are arranged in different orders by evaluating their individual discriminating ability. With experiments on different attribute ordering such as descending-order, ascending-order, original-order, and random-order, it was found that attribute ordering does have some effect on the performance and the descending-order sequence achieves the best performance. The experiments on benchmark classification problems also showed that ordering will not make a problem deceptive and will not lead to over-fitting or over-general solutions.

The results of incremental learning also hint that the performance of classifier agents may vary when running with partial attributes, which motivates us to explore the application of

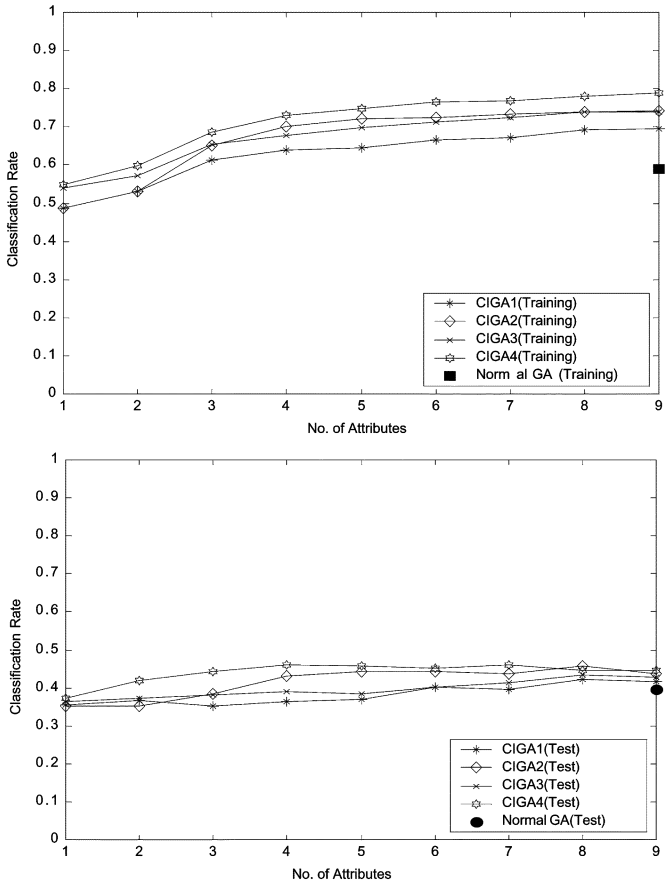


Fig. 12. Performance comparison of CIGAs on the glass data.

incremental learning for feature selection. Table VI shows the performance of the GA with some partial attributes on the diabetes data. As shown in the table, each time only one attribute is dropped. The objective is to identify which attribute is more important, and the ranking of attributes can be obtained according to the ending CR achieved. For example, we note that if attribute 2 is excluded, the ending CR from using the other seven attributes is only 0.7151, which is apparently lower than the other cases. Therefore, attribute 2 is regarded as the most important attribute. The importance of other attributes is determined accordingly and ranked. The importance of attributes in descending order is 2-3-7-6-8-4-5-1. This is a simple way to find relevant features and discarding irrelevant features.

VI. Discussions

As mentioned in Section II, there are two general approaches for the GA-based rule optimization and learning, namely, *Michigan* approach and *Pitt* approach. Our design mainly follows the *Pitt* approach, although some revisions are adopted to address the special needs of incremental learning. For instance, the addition of new attributes into chromosomes makes the length of chromosomes variable through evolution. While carrying out our work based on the *Pitt* approach, we have also made some exploration on ILGA based on the *Michigan* approach [34]. It has been found that ILGA based on the *Michigan* approach is feasible and the results were comparable (to ILGA based on the Pitts approach). The results from similar experiments on some real-world data sets showed again that

TABLE VI
Comparison of the Performance of the GA With Some Partial Attributes on the Diabetes Data

	NOA=7 (excluding attr. 1)	NOA=7 (excluding attr. 2)	NOA=7 (excluding attr. 3)	NOA=7 (excluding attr. 4)
Initial CR	0.6362 (0.0111)	0.6258 (0.0180)	0.6401 (0.0129)	0.6406 (0.0135)
Generations	188.9 (46.0)	171.1 (57.2)	165.9 (49.1)	203.1 (42.6)
T. time (s)	577.0 (146.6)	481.0 (174.7)	482.4 (150.5)	598.0 (125.8)
Train. CR	0.7706 (0.0220)	0.7151 (0.0185)	0.7547 (0.0227)	0.7667 (0.0136)
Rank	8	1	2	6
	NOA=7 (excluding attr. 5)	NOA=7 (excluding attr. 6)	NOA=7 (excluding attr. 7)	NOA=7 (excluding attr. 8)
Initial CR	0.6396 (0.0082)	0.6167 (0.0311)	0.6422 (0.0111)	0.6401 (0.0098)
Generations	191.4 (34.0)	195.1 (41.8)	197.6 (34.3)	198.8 (56.9)
T. time (s)	578.7 (114.3)	567.6 (130.1)	589.5 (103.6)	592.9 (187.3)
Train. CR	0.7690 (0.0100)	0.7641 (0.0211)	0.7607 (0.0142)	0.7661 (0.0262)
Rank	7	4	3	5

Notes:
1. mutationRate=0.01, crossoverRate=1, survivorsPercent=50%, ruleNumber=30, popSize=100, stagnationLimit=30, generationLimit=250.

incremental learning based on the *Michigan* approach achieves better performance than the retraining GA.

The main advantage of the ILGA is the initialization process, during which the obtained rule sets with a subset of attributes are effectively used when a population is initialized with the full set of attributes. The specially-designed algorithms can cater to the different initialization situations. The experimental results have shown the performance improvement advantages on the training time and classification rates. However, ILGA may be not applicable in certain situations, for example, some closely linked attributes cannot be separated into independent portions, or the nature of problem is totally changed in a dynamic environment, then the normal GA may still be a good choice.

The order of attributes, i.e., linkage of attributes, is another important issue. Holland [16] indicated that crossover induces a linkage phenomenon. It has been shown that GAs work well only if the building blocks are tightly linked on the chromosome [30]. Some algorithms have been proposed to include linkage design into problem representation and recombination operator or use some probabilistic-based models. For example, the linkage learning genetic algorithm (LLGA) was proposed in [35] for tackling the linkage and ordering problem. Several Probabilistic Model Building Genetic Algorithms (PMBGAs) have been proposed [36] to generate new child population based on probabilistic models. The dependence and linkage among the attributes in this paper also indicates possible presence of the linkage learning problem, especially when the one-point crossover operator is used. However, the proposed ILGA may alleviate linkage pressure by dividing the whole attribute domain into several parts. To find the best linkage among attributes will be a future research issue for ILGA.

There can be many variations for the algorithms and experiment settings to deal with different environmental situations. We discuss further some possibilities which include the expansion of population size to facilitate the integration of new and old elements, the algorithms for a single agent to accommodate the new attributes, and the special case when the new patterns are available with new attributes only.

As discussed earlier, before IS2 and IS4 are employed, the old and new elements have already been evolved. When they are integrated, there are many options in producing the offspring

by integration. In the current experiments, we choose the same population size for initial GAs and succeeding GAs with different initialization schemes. In order to explore more on integration, we double the population size to accommodate more resulting offspring chromosomes from the integration of old and new elements. Then the offspring chromosomes are sorted in a descending order of fitness. The fitter half will survive as the initial population for IS2 or IS4. The other half will be discarded. We have done some experiments with this refinement, and found the resulting test CR can be improved in most cases. For instance, the test CR for wine data can be improved by about 2%. As for the training time, it will be a little longer because of the longer integration process. However, it can be neglected as the following training time will dominate the whole GA process.

From the results of IS1–IS4, we find that IS2 and IS4 perform better than IS1 and IS3, mainly in reduced training time. We have considered the application of IS approaches in a multiagent environment, which means IS2 and IS4 are used only when two or more classifier agents are exchanging information on new attributes and instances. However, IS2 and IS4 can also be used in the situation of one single agent. When such an agent learns that new attributes are available, it can create the elements for these attributes first, then independently evolve these elements, as if it was done in other agents. Finally, these elements can be integrated with the old ones using IS2 or IS4. Therefore, the GAs with various initialization schemes can be refined for a single agent to achieve better performance by replacing the process of randomly creating new element with a separate evolution process for new elements before they are integrated.

In the above experiments, we assumed that the new training patterns including all attributes come along with the new attributes so that we can use these patterns to train the rule set with all attributes. This is likely in most realistic applications. For instance, a researcher may find a new symptom which is likely to contribute to the diagnosis of a certain disease. Then, new data associated with the old and new symptoms may be collected for further research. However, sometimes newly-collected data may only contain information on the new attributes, without information on the old attributes. Therefore, when the old and new attributes are integrated, there is no hands-on training patterns to train the rule set for the whole set of attributes. We will need to integrate the old and new training patterns together to form a new training pattern set according to the class categories. Then, classifier agents can use ILGA with different IS approaches to evolve a new rule set on the new training patterns.

VII. Conclusions and Future Work

In this paper, we have proposed an incremental approach to the GA, with different initialization schemes for incremental learning in classifier agents. These approaches do not need to re-evolve the rule set from scratch in order to adapt to the ever-changing environment. Using these approaches, a classifier agent can fully utilize existing knowledge and quickly respond to changes in the environment.

The main features such as the incremental evolution of new attributes, formation of new population, integration of chromosomes, and biased rates for genetic operators were elaborated. Real-world data sets were used to evaluate the performance of the incremental approach. The experimental results showed that GAs with IS approaches can be successfully used for incremental learning and may generally speed up the learning process and improve classification rates as compared to the retraining GA. Possible applications for continuous incremental training and feature selection were also presented.

In our experiments, we assumed that the incoming attributes are relevant to the old attributes. In a real-world situation, classifier agents may need the ability to identify whether the incoming new attributes data are consistent with the old attributes by observing the trend of training or test performance. When agents exchange attributes, classes, or training data, some overlapping can happen. For example, the training data in different agents may be identical or nonidentical, thus the exchanged training data may have different degrees of overlap. Therefore, a classifier agent needs to analyze the incoming data, and integrate them with its own current data using certain methods. Furthermore, if elements/rules exchanged among agents have some overlapping, agents may need more advanced mechanisms to detect them and integrate them properly.

In our experiments, we set the maximum number of rules empirically. Actually, agents can increase the rule number gradually until they find there is no improvement on the classification rate. However, in this way, training time may become unacceptable. It can be turned into a multiobjective problem, as we want to balance among rule number, classification rate, and training time.

Appendix I
Genetic Operators

We use one-point crossover in all experiments. Referring to the encoding mechanism, we can note that crossover will not lead to inconsistency and thus can take place in any point of chromosome. On the contrary, the mutation operator has some constraints. The mutation point is randomly selected. According to the position of selected point, we can determine whether it is an activeness, minimum, or maximum element. Different mutation is available for each. For example, if an activeness element is selected for mutation, it will just be toggled. Otherwise when a boundary-value element is selected, the algorithm will randomly select a substitute in the range of that attribute. This is implemented in such a way that the lower and upper bounds are never exceeded. The mutation and crossover rates are selected as 0.01 and 1.0 respectively (mutationRate $= 0.01$ and crossoverRate $= 1.0$). For reproduction, we simply set the survival rate (or generation gap) as 50% (SurvivorsPercent $= 50\%$), which means half of the parent chromosomes with higher fitness will survive into the new generation, while the other half will be replaced by the newly created children resulting from crossover and/or mutation.

Selection mechanism deals with the selection of a population which will undergo genetic operations. Roulette wheel selection [37] is used in this paper. In this investigation, the probability that a chromosome will be selected for integration is given by the chromosome's fitness divided by the total fitness of all the

```
IS1 :
  for each newChrom[j] in the new population
    {bestChrom := the best chromosome from the current solution;
     bufferChrom := bestChrom;
     for each rule i in bufferChrom
       {randomly create the activeness bit and bounds for each new attribute;
        create each element with the activeness bit and boundary values;
        insert all the created elements into bufferChrom;
       }
     newChrom[j] := bufferChrom;
    }
```

```
IS3 :
  for each newChrom[j] in the new population
    {bufferChrom := the chromosome[j] from the current solution;
     for each rule i in bufferChrom
       {randomly create the activeness bit and bounds for each new attribute;
        create each element with the activeness bit and boundary values;
        insert all the created elements into bufferChrom;
       }
   newChrom[j] := bufferChrom;
  }
```

```
IS4 :
  for each newChrom[j] in the new population
    {bufferChrom := the chromosome[j] from the current solution;
     incomingChrom := a chromosome randomly selected from the group
          chromosomes coming from another agent;
     for each rule i in bufferChrom
       {curClass := the class of rule i;
        analyze incomingChrom, and place all new incoming rules having
          the same class as curClass into a candidate pool;
        randomly choose a rule from the candidate pool;
           insert all the elements for the new attributes in the selected rule
           into bufferChrom;
       }
     newChrom[j] := bufferChrom;
    }
```

chromosomes. This means, during selection, higher fitness chromosomes have a higher probability of producing offspring for the next generation than lower fitness chromosomes.

APPENDIX II
PSEUDOCODE FOR THE FORMATION OF A NEW POPULATION USING IS1, IS3, AND IS4

See the table at the top of the page.

ACKNOWLEDGMENT

The authors would like to thank the anonymous referees for the constructive comments. The second author is grateful to the Singapore Millennium Foundation for the fellowship awarded.

REFERENCES

[1] C. Giraud-Carrier, "A note on the utility of incremental learning," *AI Commun.*, vol. 13, no. 4, pp. 215–223, 2000.
[2] J. M. Bradshaw, *Software Agent*. Cambridge, MA: MIT Press, 1997.
[3] S. M. Weiss and C. A. Kulikowski, *Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems*. San Mateo, CA: Morgan Kaufmann, 1991.
[4] K. Yamauchi, N. Yamaguchi, and N. Ishii, "Incremental learning methods with retrieving of interfered patterns," *IEEE Trans. Neural Netw.*, vol. 10, no. 6, pp. 1351–1365, Nov. 1999.
[5] S. U. Guan and S. Li, "Incremental learning with respect to new incoming input attributes," *Neural Process. Lett.*, vol. 14, no. 3, pp. 241–260, 2001.
[6] L. Su, S. U. Guan, and Y. C. Yeo, "Incremental self-growing neural networks with the changing environment," *J. Intell. Syst.*, vol. 11, no. 1, pp. 43–74, 2001.
[7] K. A. DeJong and W. M. Spears, "Learning concept classification rules using genetic algorithms," in *Proc. 1991 Int. Joint Conf. Artificial Intelligence*, 1991, pp. 651–656.

[8] J. J. Merelo, A. Prieto, and F. Moran, "Optimization of classifiers using genetic algorithms," in *Advances in the Evolutionary Synthesis of Intelligent Agents*, M. Patel, V. Honavar, and K. Balakrishnan, Eds. Cambridge, MA: MIT Press, 2001.
[9] H. Ishibuchi, T. Nakashima, and T. Murata, "Performance evaluation of fuzzy classifier systems for multidimensional pattern classification problems," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 29, no. 5, pp. 601–618, Oct. 1999.
[10] H. Ishibuchi and 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.
[11] H. Kang, "Evolvable cellular classifiers," in *Proc. Congr. Evolutionary Computation*, 2000, pp. 464–470.
[12] M. Setnes and H. Roubos, "GA-Fuzzy modeling and classification: Complexity and performance," *IEEE Trans. Fuzzy Syst.*, vol. 8, no. 5, pp. 509–522, Oct. 2000.
[13] D. B. Fogel, L. J. Fogel, and J. W. Atmar, "Meta-evolutionary programming," in *Proc. 25th Aslimar Conf. Signals, Systems and Computers*, 1991, pp. 540–545.
[14] J. R. Koza, *Genetic Programming*. Cambridge, MA: MIT Press, 1992.
[15] H. P. Schwefel and G. Rudolph, "Contemporary evolution strategies," in *Proc. 3rd Int. Conf. Artificial Life*, vol. LNAI 929, 1995, pp. 893–907.
[16] J. H. Holland, *Adaptation in Nature and Artificial Systems*. Ann Arbor, MI: Univ. Michigan Press, 1975.
[17] P. L. Lanzi, W. Stolzmann, and S. W. Wilson, *Learning Classifier Systems: From Foundations to Applications*. Berlin, Germany: Springer, 2000.
[18] A. L. Corcoran and S. Sen, "Using real-valued genetic algorithm to evolve rule sets for classification," in *Proc. 1st IEEE Conf. Evolutionary Computation*, Orlando, FL, Jun. 1994, pp. 120–124.
[19] L. Fu, H. Hsu, and J. C. Principe, "Incremental backpropagation learning networks," *IEEE Trans. Neural Netw.*, vol. 7, no. 3, pp. 757–761, May 1996.
[20] R. Polikar, L. Udpa, S. S. 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.
[21] F. Dalché-Buc and L. Ralaivola, "Incremental learning algorithms for classification and regression: Local strategies," in *Proc. American Institute of Physics Conf.*, vol. 627, 2001, pp. 320–329.
[22] F. S. Osorio and B. Amy, "INSS: A hybrid system for constructive machine learning," *Neurocomputation*, vol. 28, pp. 191–205, 1999.
[23] A. P. Engelbrecht and R. Brits, "A clustering approach to incremental learning for feedforward neural networks," in *Proc. Int. Joint Conf. Neural Network*, vol. 3, 2001, pp. 2019–2024.
[24] J. H. Holland, "Escaping brittleness: The possibilities of general purpose learning algorithms applied to parallel rule-based systems," in *Machine Learning: An Artificial Intelligence Approach*, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, Eds. Los Altos, CA: Morgan Kaufmann, 1986.
[25] K. A. DeJong, "Learning with genetic algorithms: An overview," *Mach. Learn.*, vol. 3, pp. 121–138, 1988.
[26] S. F. Smith, "A Learning System Based on Genetic Adaptive Algorithms," Ph.D. dissertation, Univ. Pittsburgh, Pittsburgh, PA, 1980.
[27] E. Bernado, X. Llora, and J. M. Garrell, "XCS and GALE: A comparative study of two learning classifier systems on data mining," in *Int. Workshop Learning Classifier Systems*, vol. LNAI 2321, 2002, pp. 115–132.
[28] G. Enee and C. Escazut, "Classifier systems evolving multi-agent system with distributed elitism," in *Proc. 1999 Congr. Evolutionary Computation*, 1999, pp. 1740–1746.
[29] D. Caragea, A. Silvescu, and V. Honavar, "Toward a theoretical framework for analysis and synthesis of distributed and incremental learning agents," in *Proc. Workshop on Distributed and Parallel Knowledge Discovery*, Boston, MA, 2000.
[30] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.
[31] C. L. Blake and C. J. Merz. (1998) UCI Repository of Machine Learning Databases. Dept. Inform. and Comput. Sci., Univ. California, Irvine. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html
[32] L. Prechelt, "PROBEN1: A set of Neural Network Benchmark Problems and Benchmarking Rules," Dept. Informatics, Univ. Karlsruhe, Karlsruhe, Germany, Tech. Rep. 21/94, 1994.
[33] F. Zhu and S. U. Guan, "Ordered incremental training with genetic algorithms," *Int. J. Intell. Syst.*, vol. 19, no. 12, pp. 1239–1256, 2004.
[34] C. Y. Cheah, "Incremental Genetic Algorithm With the Real Michigan-Style Classifier," B.Eng. thesis, National Univ. Singapore, 2004.
[35] G. R. Harik and D. E. Goldberg, "Learning linkage through probabilistic expression," in *Comput. Methods in Appl. Mechan. and Eng.*, 2000, vol. 186, pp. 295–310.
[36] M. Pelikan, D. E. Goldberg, and E. Cantú-paz, "Linkage problem, distribution estimation and bayesian networks," *Evolut. Comput.*, vol. 8, no. 3, pp. 311–340, 2000.
[37] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd ed. New York: Springer-Verlag, 1996.

Sheng-Uei Guan received the M.Sc. and Ph.D. degrees from the University of North Carolina, Chapel Hill.

He is currently with the Electrical and Computer Engineering Department, National University of Singapore. He previously worked in a prestigious R&D organization for several years, serving as a Design Engineer, Project Leader, and Manager. He has served as a member of the R.O.C. Information and Communication National Standard Draft Committee. After leaving the industry, he joined Yuan-Ze University, Taiwan, R.O.C., for over three years, where he served as Deputy Director for the Computing Center, and also as the Chairman for the Department of Information and Communication Technology. Later, he joined the Department of Computer Science and Computer Engineering, La Trobe University, Melbourne, Australia, where he helped to create a new multimedia systems stream.

Fangming Zhu received the Ph.D. degree from the National University of Singapore in 2004, and the B.Eng. and M.Eng. degrees from the Shanghai Jiaotong University, Shanghai, China, in 1994 and 1997, respectively.

He is currently a Singapore Millennium Foundation Postdoctoral Fellow in the Department of Electrical and Computer Engineering, National University of Singapore. His current research interests include evolutionary computation, pattern classification, and intelligent agents.