Ordered Incremental Training with Genetic Algorithms

Fangming Zhu,† Sheng-Uei Guan*
Department of Electrical and Computer Engineering, National University of Singapore, 10 Kent Ridge Crescent, Singapore 119260

Incremental training has been used for genetic algorithm (GA)-based classifiers in a dynamic environment where training samples or new attributes/classes become available over time. In this article, ordered incremental genetic algorithms (OIGAs) are proposed to address the incremental training of input attributes for classifiers. Rather than learning input attributes in batch as with normal GAs, OIGAs learn input attributes one after another. The resulting classification rule sets are also evolved incrementally to accommodate the new attributes. Furthermore, attributes are arranged in different orders by evaluating their individual discriminating ability. By experimenting with different attribute orders, different approaches of OIGAs are evaluated using four benchmark classification data sets. Their performance is also compared with normal GAs. The simulation results show that OIGAs can achieve generally better performance than normal GAs. The order of attributes does have an effect on the final classifier performance where OIGA training with a descending order of attributes performs the best. © 2004 Wiley Periodicals, Inc.

1. INTRODUCTION

Classification problems play a major role in various fields of computer science and engineering, such as image processing and data mining. A number of soft computing approaches, such as neural networks,[1,2,3] evolutionary algorithms,[4,5] and fuzzy logic,[6,7] have been widely used to evolve solutions adaptively for classification problems. Among them, genetic algorithms (GAs) have attracted much attention and become one of the most popular techniques for classification.[4,8]

Rule-based solution is widely used in GAs for classification problems, either through supervised or unsupervised learning.[9] Conventionally, most GA-based work

*Author to whom all correspondence should be addressed: e-mail: eleguans@nus.edu.sg.
†e-mail: elezfm@nus.edu.sg.

INTERNATIONAL JOURNAL OF INTELLIGENT SYSTEMS, VOL. 19, 1239–1256 (2004)
 Published online in Wiley InterScience (www.interscience.wiley.com). • DOI 10.1002/int.20046

in the literature concentrates on batch-mode, static domain, where the attributes, classes, and training data are all determined a priori and the task of GA is to find out the best rule sets that classify the available instances with the lowest error rate.[4,6] However, some learning tasks do not fit into this static model. As the real-world situation is more dynamic and keeps changing, a classifier is actually exposed to the changing environment, and it needs to evolve its solutions by adapting to various changes. Classifiers may learn the solution from scratch, that is, from nil initial knowledge, or revise the current solutions to incorporate new knowledge and adapt to the changing environment. For example, new attributes may be found as possible contributors for the classification problem. Then, the old solution can be evolved to incorporate these new attributes, which demands some algorithms to achieve this objective.

Many researchers have addressed incremental training algorithms and methods. Yamauchi et al.[10] proposed incremental learning methods for retrieving interfered patterns. In their methods, a neural network learns new patterns with relearning of a small number of retrieved past patterns that interfere with the new patterns. Fu et al.[11] 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. Polikar et al.[12] introduced Learn++, an algorithm for incremental training of neural networks. Guan and Li[2] presented five incremental learning algorithms to retain the existing neural network and train a new subnetwork incrementally when new attributes come in. The new subnetwork and the old one are merged later to form a new network. Su et al.[3] proposed an adaptive incremental learning mechanism, which employs pyramid-tower neural networks for automatic network structure design and exploits information learned by the previously grown network to avoid retraining. However, most work explores incremental learning with statistical algorithms or neural networks, and few touch on the use of evolutionary algorithms.

This article uses the incremental genetic algorithm (IGA) as a basic evolutionary algorithm for incremental learning,[13,14] and proposes ordered incremental genetic algorithms (OIGAs) for the incremental training of input attributes for classifiers. Different from normal GAs, which learn input attributes in their full dimension, OIGAs learn the attributes one after another under a situation of continuous incremental learning. Each time when learning a new attribute, OIGAs keep the old solution and integrate it with the next new attribute to evolve further into a new solution. Furthermore, attributes are arranged in different orders by evaluating their individual discriminating ability. By experimenting with different attribute orders, different approaches of OIGAs are evaluated. The simulation results show that OIGAs can achieve generally better performance than normal GAs. The order of attributes does have an effect on the final classifier performance where OIGA training with a descending order of attributes performs the best.

The rest of the article is organized as follows. In Section 2, the algorithm of IGA is introduced. Then, the details of OIGAs are elaborated in Section 3. The experimental results of OIGAs on four benchmark data sets and their analysis are reported in Section 4. Section 5 presents some discussion based on the experimental results, and Section 6 concludes the article.

```
Loop begin
if a new attribute is ready to be integrated
   t:=0;
   select group chromosomes in the old solution as seeds;
   integrate each old chromosome with the new elements for new attributes to form a
         new population P(t);
   evaluate P(t);
   while (not terminate-condition) do       //stopping criteria
         begin
            select P'(t) from P(t);   // based on roulette wheel selection
            crossover P'(t);
            mutation P'(t);
            combine P'(t) and P(t) to form P(t+1);
            evaluate P(t+1);
            t:=t+1;
         end
Loop end
```

Figure 1. Pseudocode of IGA.

2. INCREMENTAL GENETIC ALGORITHM (IGA)

An incremental genetic algorithm (IGA) is proposed for incremental learning of new attributes/classes for classifiers.[13,14] It is developed on the basis of a normal GA. Figure 1 illustrates the pseudocode of IGA. When a new attribute is ready to be integrated, an initial population is formed by integrating the old chromosomes (solutions) and new elements for the new attribute if available. Based on fitness, some chromosomes are selected with a roulette wheel selection. Crossover and mutation will be then applied on these selected chromosomes and the child population is thus generated. A certain percentage of the parent population will be preserved and the rest will be replaced by the child population. The evolution process will continue until it satisfies the stopping criteria.

2.1. Encoding Mechanism

We use the non-fuzzy IF-THEN rules with continuous attributes for classifiers. A rule set consisting of a certain number of rules is a solution candidate for a classification problem.

An IF-THEN rule is represented as follows:

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

where R_i is a rule label, n is the number of attributes, $(x_1, x_2, \ldots, x_n)$ is the input attribute set, and y is the output class category assigned with a value of C. V_{jmin} and V_{jmax} are the minimum and maximum bounds of the jth attribute x_j, respectively. We encode rule R_i according to the diagram shown below:

Antecedent element 1			...	Antecedent element n			Consequence element
Act_1	V_{1min}	V_{1max}	...	Act_n	V_{nmin}	V_{nmax}	C

where Act_j denotes whether condition j is active or inactive, which is encoded as 1 or 0.
Note: If V_{jmin} is larger than V_{jmax} at any time, this element will be regarded as an invalid element. Invalid elements will make no contribution in the classification rule.

Each antecedent element represents an attribute, and the 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:

$$CR_j = \bigcup_{i=1,m} R_i \quad j = 1,2,\ldots,s \tag{2}$$

where m is the maximum number of rules allowed for each chromosome, and s is the population size. Therefore, one chromosome will represent one rule set. Because we know the discrete value range for each attribute and class a priori, V_{jmin}, V_{jmax}, and C can be encoded each as a character by finding their positions in the ranges. Thus, the final chromosome can be encoded as a string.

2.2. Genetic Operators

One-point crossover is used in this article. It can take place anywhere in a chromosome. Referring to the encoding mechanism, as all chromosomes have the same structure, the crossover of two chromosomes will not cause inconsistency. On the contrary, the mutation operator has some constraints. Different mutation is available for different elements. 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. The rates for mutation and crossover are selected as 0.01 and 1.0.

We set the survival rate as 50% (SurvivorsPercent = 50%), which means half of the parent chromosomes with higher fitness will survive into the new generation, whereas the other half will be replaced by the newly created children resulting from crossover and/or mutation. Roulette wheel selection is used in this article as the selection mechanism.[15] In this investigation, the probability that a chromosome will be selected for mating is given by the chromosome's fitness divided by the total fitness of all the chromosomes.

2.3. Fitness Function

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.

Because there is more than one rule in a chromosome, it is possible that there can be multiple rules matching the conditions for all the attributes but predicting

different classes. We use a voting mechanism to help resolve any 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 means that this instance cannot be classified correctly by this rule set.

2.4. Stopping Criteria

There are three factors in the stopping criteria. The evolution process stops after a preset generation limit, or when the best chromosome's fitness reaches a preset threshold (which is set as 1.0 through this article), or when the best chromosome's fitness has shown no improvement over a specified number of generations—stagnationLimit. The detailed settings are reported along with the corresponding results.

2.5. Initial Population for IGA

The formation of the initial population is one of the main features of IGA, in which the integration of old chromosomes with new elements is performed. Figure 2 shows how the new element for a new attribute is 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.

Figure 3a illustrates the formation of a new population under IGA and Figure 3b shows the corresponding pseudocode of how new chromosomes are created by integrating old chromosomes with new elements. We can see that IGA copies the group chromosomes into the new chromosomes, and new elements are integrated with the old chromosomes with a class-matching mechanism as explained in Figure 3b.

3. ORDERED INCREMENTAL GENETIC ALGORITHMS (OIGAs)

Let us assume a classification problem has c classes in the n-dimensional pattern space, and p vectors $X_i = (x_{i1}, x_{i2}, \ldots, x_{in})$, $i = 1,2,\ldots,p$, $p \gg c$, are given as training patterns. The task of classification is to assign instances to one out of a set of predefined classes, by discovering certain relationship among

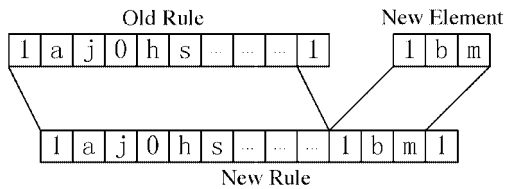
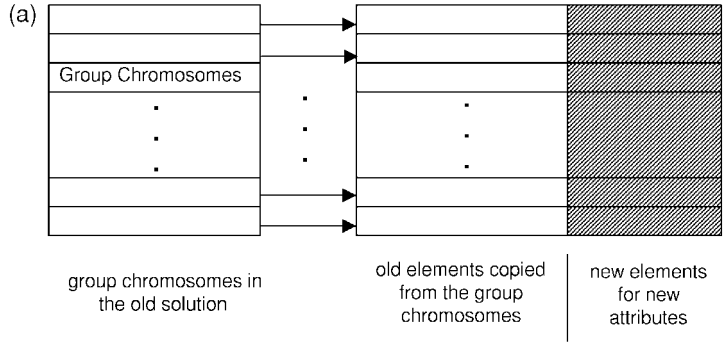


Figure 2. Formation of a new rule in a new chromosome.



(b)

```
for each newChrom[j] in the new population for IGA
 { bufferChrom:=the chromosome[j] from the last generation of GA/IGA;
   incomingChrom:=a chromosome randomly selected from the available new group
                  chromosomes;
   for each rule i in bufferChrom
        { curClass:=the class of rule i;      //class matching mechanism
        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;
 }
```

Figure 3. (a) Illustration for forming a new population under IGA. (b) Pseudocode for integrating old chromosomes with new elements.

attributes. Then, the discovered rules can be evaluated by classification accuracy or error rate either on the training data or test data.

Figure 4 illustrates the concepts of normal GAs and OIGAs. As shown in Figure 4a, a normal GA maps attributes to classes directly in a batch manner, which means all the attributes, classes, and training data are used together to train a group of GA chromosomes. OIGAs are significantly different. As shown in Figure 4b, they consist of several stages, each stage employing an IGA to accommodate a new attribute. They may start from nil knowledge. With the continuous introduction of new attributes, the rule set can be incrementally evolved step by step and achieve the final solution. OIGAs can also stop at any intermediate stage, and the solution at that time can be used to tackle the problem with a subset of attributes.

The single-attribute evolution modules (SEMs) are used to separately evolve a single attribute, and the resultant group chromosomes are used as new elements for the new attribute in IGA (cf. Figures 1 and 3b). Considering that it is relatively easy to evolve a single attribute, we choose the parameters for SEMs as half of the values used in the normal iteration to reduce the time cost. This means generationLimit, ruleNumber, and popSize are set as half of the normal iteration values.

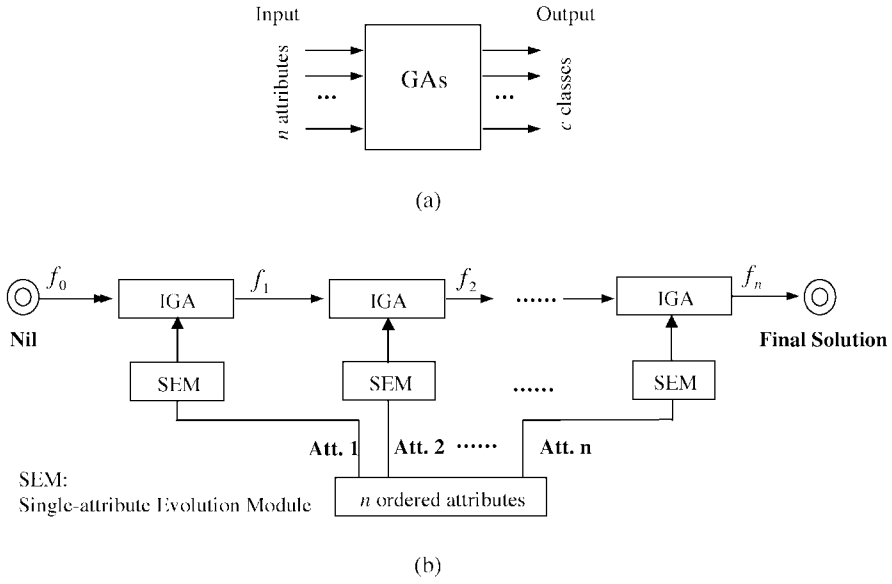


Figure 4. Illustrations of normal GAs and OIGAs: (a) normal GA-based solution and (b) illustration of OIGAs.

Following the notations presented above, we denote each stage of OIGAs as

$$f_i : (f_{i-1}, X_i) \rightarrow C \quad i = 1,2,\ldots,n \tag{3}$$

where f_i is a solution for the problem with i attributes, and f_0 is a special case—nil solution. X_i is the vector of training patterns with i attributes, and C is the set of output classes. In each stage i, X_{i-1} is expanded into X_i by introducing the new ith attribute, and the old solution f_{i-1} is also used to help evolve a new solution f_i. Therefore, $(f_0, f_1, \ldots, f_n)$ provides the corresponding solutions with the incremental introduction of new attributes, and f_n is the final solution accommodating the whole set of attributes.

Figure 5 shows the algorithms for OIGAs. Note that the function of SEMs is fulfilled in step 5. In step 6, the new elements obtained from step 5 will be integrated with the old chromosomes, as presented in Section 2.5. Step 1 and step 2 in OIGAs are used to select the order of attributes for incremental training. Each attribute will be evaluated, and its individual discriminating ability is measured by the training and test classification rates achieved. Both rates can be used as the metric. In this article, the average of these two rates is used as the metric. We have some further discussions on this issue in Section 5. With the individual discriminating ability of each attribute computed, there are a number of options for the order of attributes available for incremental training, considering the different sequence combinations of attributes. In this article, four choices are selected for experiments, namely, descending order, ascending order, original order, and random order. As indicated by their names, each approach arranges attributes in a

Step 1: Evaluate the individual discriminating ability of each attribute.

Step 2: Order the attributes according to the selected approach.

Step 3: Initialization. Set i equal to 1. Initialize parameters for OIGAs.

Step 4: i-th iteration. Introduce the i-th attribute.

Step 5: Evolve the i-th attribute alone in SEM, using the training patterns for the i-th attribute.

Step 6: IGA is used to integrate the new elements for the i-th attribute, forming the new initial population. (i=1 means the introduction of the first attribute. As there is no old solution, the new elements for the first attribute are simply used as the initial population.)

Step 7: Evolve the chromosomes until the stopping criteria are met, using the training patterns including attributes from the 1st to the i-th.

Step 8: If i is equal to n (which is the total number of attributes), the training process will be stopped. Otherwise, increase i by 1. Repeat the procedure from step 4 to step 8.

Figure 5. Algorithms for OIGAs.

specific order. Descending order means attributes are arranged in a descending manner according to their discriminating ability, whereas in ascending order, attributes are arranged in reverse order. Original order maintains the same order as those in the original data set, whereas random order arranges attributes randomly. After the order of attributes is selected, attributes will be presented for incremental training in the prescribed sequence.

4. EXPERIMENTS AND ANALYSIS

We have implemented several classifiers running on four benchmark data sets, which are the yeast data, glass data, wine data, and cancer data. They all are real-world problems, and are available in the UCI machine learning repository.[16] Each data set is equally partitioned into two parts. One half is for training and the other half is for testing.

All experiments are completed on Pentium III 650-MHz PCs with 128 MB memory. The results reported are all averaged over 10 independent runs. The

parameters, such as mutation rate, crossover rate, generation limit, stagnation limit, and so on, are given under the results. We record the evolution process by noting down some indicative results, which include initial classification rate (CR), generation cost, training time, training CR, and test CR. (Their exact meanings can be found in the notes under Table I.) Only one attribute is introduced during each iteration in OIGAs, and the sequence of introducing attributes is determined by the order selected.

The yeast problem predicts the protein localization sites in cells. It has 8 attributes, 9 classes, and 1484 instances. Table I shows the individual discriminating ability of each attribute of the yeast data. Each column records an independent evolution process for each attribute. It has an initial CR that is the initial classification rate achieved on the training data with the randomly generated population. After some generations (training time), the classifier achieves an ending CR on the training data and a test CR on the test data. As explained earlier, the attributes are ranked according to the averaged values of training and test CRs. Using this rank, the attribute sequence for the four approaches of OIGAs can be determined accordingly.

The classifier performance on the yeast data with the four approaches is shown in Table II. For each approach, the whole learning process is recorded. It consists of several columns, each of which corresponds to one iteration in the algorithm for OIGAs (cf. Figure 5). The bottom part of Table II provides a summary for all OIGAs. The generations and training time consumed in all stages are summed up. The initial CR in the summary table is actually the initial CR with the first attribute, which is different from the initial CR for a normal GA (with the whole set of attributes). The ending CR and test CR are the final training and test CRs with the whole attribute set introduced. They are also compared to normal GA with the improvement percentage computed.

Table I. Individual discriminating ability of each attribute of the yeast data.

	Att. 1	Att. 2	Att. 3	Att. 4	Att. 5	Att. 6	Att. 7	Att. 8
Initial CR	0.2595	0.2723	0.2903	0.2636	0.2898	0.2579	0.2817	0.2656
Generations	55.5	55	52.2	55.3	33.2	39.4	38.1	43.2
T. time (s)	139.1	138.2	130.8	135.4	90.9	89.2	96.3	105.5
Ending CR	0.3089	0.3078	0.3724	0.3625	0.2906	0.3005	0.2896	0.3023
Test CR	0.3065	0.2972	0.368	0.3452	0.2893	0.2925	0.2879	0.2929
Ave. CR	0.3077	0.3025	0.3702	0.3539	0.29	0.2965	0.2888	0.2976
Rank	3	4	1	2	7	6	8	5

mutationRate = 0.01, crossoverRate = 1, survivorsPercent = 50%, ruleNumber = 30, popSize = 100, stagnationLimit = 30, generationLimit = 60.
Initial CR: the best classification rate achieved by the initial population; Generations: the generation needed to reach the stopping criteria; T. time (s): the training time cost, and its unit is second; Ending CR: the best classification rate achieved by the resulting population on the training data; Test CR: the classification rate achieved on the test data.
The other tables follow the same notations as this table.

Table II. Performance comparison on the yeast data—OIGAs and normal GA.

OIGA_Desc	Add Att. 3	Add Att. 4	Add Att. 1	Add Att. 2	Add Att. 8	Add Att. 6	Add Att. 5	Add Att. 7
Initial CR	0.2714	0.3691	0.4024	0.4061	0.4198	0.4259	0.4477	0.4415
Generations	55.2	53.5	51.9	52.9	55.7	58.7	46.8	50.7
T. time (s)	147.8	217.3	238.2	273.5	311.5	347.7	303.2	324.5
Ending CR	0.3721	0.4096	0.4181	0.4255	0.4416	0.4508	0.4559	0.4573
Test CR	0.365	0.3918	0.3961	0.3989	0.407	0.4128	0.4131	0.4135
OIGA_Asce	Add Att. 7	Add Att. 5	Add Att. 6	Add Att. 8	Add Att. 2	Add Att. 1	Add Att. 4	Add Att. 3
Initial CR	0.2644	0.2898	0.2943	0.2996	0.3009	0.3086	0.3237	0.3636
Generations	34.7	33.4	45.8	39.5	50.8	53.8	54.4	57.3
T. time (s)	94.3	161.5	234.2	228.5	311.2	343.9	373.9	399.5
Ending CR	0.2892	0.2902	0.2999	0.3009	0.3125	0.3233	0.3574	0.3919
Test CR	0.2881	0.2885	0.2923	0.2923	0.2965	0.3074	0.3363	0.3609
OIGA_Orig	Add Att. 1	Add Att. 2	Add Att. 3	Add Att. 4	Add Att. 5	Add Att. 6	Add Att. 7	Add Att. 8
Initial CR	0.2612	0.3061	0.3333	0.3763	0.4111	0.4046	0.4221	0.4129
Generations	58.2	50.1	57.7	53.9	50.3	56.1	50.8	52.4
T. time (s)	175.5	245	319.5	338.8	353.9	410.5	401.4	410.9
Ending CR	0.309	0.3171	0.385	0.4144	0.422	0.427	0.4286	0.4326
Test CR	0.3073	0.3108	0.3677	0.39	0.3911	0.3935	0.3943	0.396
OIGA_Rand	Add Att. 2	Add Att. 1	Add Att. 4	Add Att. 3	Add Att. 7	Add Att. 5	Add Att. 6	Add Att. 8
Initial CR	0.254	0.3075	0.3349	0.3728	0.3992	0.4228	0.4182	0.4186
Generations	52.9	56	58.3	58.3	56.1	57.3	54.6	53.1
T. time (s)	139.9	236.5	282.1	320.1	336	373.6	375.4	373.8
Ending CR	0.3098	0.3228	0.3755	0.4174	0.424	0.4341	0.4345	0.435
Test CR	0.2995	0.31	0.353	0.385	0.3883	0.391	0.392	0.3953

Summary	Normal GA	OIGA_Desc	OIGA_Asce	OIGA_Orig	OIGA_Rand
Initial CR	0.2127	0.2714	0.2644	0.2612	0.254
Generations	111.7	425.4	369.7	429.5	446.6
T. time (s)	628.2	2163.7	2147	2655.5	2437.4
Ending CR	0.3414	0.4573 (33.9%)	0.3919 (14.8%)	0.4326 (26.7%)	0.435 (27.4%)
Test CR	0.3284	0.4135 (25.9%)	0.3609 (9.9%)	0.396 (20.6%)	0.3953 (20.4%)

mutationRate = 0.01, crossoverRate = 1, survivorsPercent = 50%, ruleNumber = 30, popSize = 100, stagnationLimit = 30.
For each stage of OIGAs, generationLimit = 60; for normal GA, generationLimit = 480. (To be fair, the generationLimit for normal GA is equal to the sum of those for each stage. The other experiments in this article follow the same mechanism.)

Figure 6 shows the performance comparison among OIGAs and a normal GA on the yeast data in terms of training and test CRs. As a normal GA uses the whole set of attributes in batch, there are only two points in the figure. Conversely, OIGAs each have a curve to show their incremental training process. From Table II and Figure 6, it is found that the training and test CRs rise gradually with the introduction of attributes for all OIGAs. We also find that all OIGAs outperform a normal GA in terms of training CR and test CR, with a significant improvement around 10%–30%. Among the four approaches with different attribute orders, OIGA-Desc achieves the best performance, OIGA-Asce achieves the worst, and OIGA-Orig and OIGA-Rand lie in the middle.

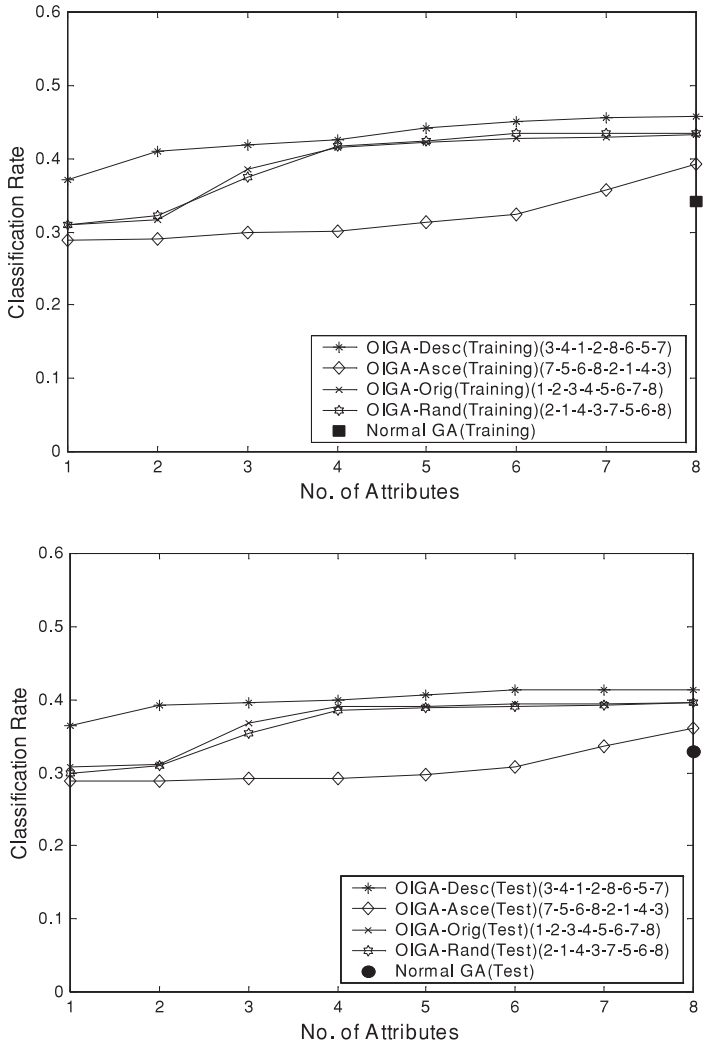


Figure 6. Comparison of OIGAs and normal GA on the yeast data.

The glass data set contains data of different glass types. The results of chemical analysis of glass splinters (the percentage of eight different constituent elements) plus the refractive index are used to classify a sample to be either float processed or nonfloat processed building windows, vehicle windows, containers, tableware, or head lamps. This data set consists of 214 instances with 9 continuous attributes from 6 classes.

Using the same method as the yeast data, Table III reports the individual discriminating ability of each attribute of the glass data. With the resultant rank, the four approaches of OIGAs can be formulated. Using different sequences of attributes presented for incremental training, the classifier is trained in different courses. The

Table III. Individual discriminating ability of each attribute of the glass data.

	Att. 1	Att. 2	Att. 3	Att. 4	Att. 5	Att. 6	Att. 7	Att. 8	Att. 9
Initial CR	0.3678	0.3505	0.3502	0.3624	0.3479	0.3537	0.3544	0.3341	0.3467
Generations	59.2	53.8	57.6	57.5	49.4	52.7	58.1	49.3	47.3
T. time (s)	58.9	53.5	58.1	57	49.3	50.1	57.3	46.7	45.2
Ending CR	0.5558	0.4311	0.5266	0.5306	0.3902	0.4089	0.4801	0.4736	0.3939
Test CR	0.3722	0.4484	0.3764	0.4619	0.3301	0.372	0.31	0.4682	0.3467
Ave. CR	0.464	0.4398	0.4515	0.4963	0.3602	0.3905	0.3951	0.4709	0.3703
Rank	3	5	4	1	9	7	6	2	8

mutationRate = 0.01, crossoverRate = 1, survivorsPercent = 50%, ruleNumber = 30, popSize = 200, stagnationLimit = 30, generationLimit = 60.

performance comparison in Table IV illustrates the whole training process and provides a summary of comparison. Figure 7 shows the performance comparison among OIGAs and a normal GA in terms of training and test CRs visually. It is found that, for the glass data, OIGAs outperform a normal GA in terms of training and test CRs. Among the four OIGA approaches, OIGA-Desc still outperforms the other OIGAs.

More experiments are conducted with OIGAs on the wine and cancer data. The wine data contains the chemical analysis of 178 wines from three different cultivars in the same region in Italy. The analysis determines the quantities of 13 constituents found in each of the three types of wines. In other words, it has 13 continuous attributes, 3 classes, 178 instances. The cancer problem diagnoses whether a breast cancer is benign or malignant. It has 9 attributes, 2 classes, and 699 instances. Due to the limited space, only the summary of performance comparison for the two data sets is listed, and the individual discriminating ability and detailed information of each stage of OIGAs are omitted.

Table V and Figure 8 show the performance comparison on the wine data, using OIGAs and a normal GA. The sequence of attributes for each OIGA approach is shown in the legend of Figure 8. It is found again that OIGAs generally outperform a normal GA in terms of training CR and test CR, although the improvement is small, especially for the training CR. Furthermore, OIGA-Desc outperforms the other approaches in terms of both training and test CRs. This result conforms to those obtained from the experiments for the above data sets.

Table VI and Figure 9 compare the performance of four types of OIGAs on the cancer data in two aspects, training CR and test CR. With a comparison among OIGAs and normal GA, we find that the training and test CRs for these approaches are similar, with not much improvement or deterioration. Figure 9 also shows visually that their performance is very close, and the OIGA-Desc performs the best again with its curve lying above the other curves.

5. DISCUSSION

From the experiments on the four benchmark data sets, we have an overall picture on OIGAs. First, with OIGAs, classifiers are equipped with a capability

Table IV. Performance comparison on the glass data—OIGAs and normal GA.

OIGA_Desc	Add Att. 4	Add Att. 8	Add Att. 1	Add Att. 3	Add Att. 2	Add Att. 7	Add Att. 6	Add Att. 9	Add Att. 5
Initial CR	0.3579	0.5888	0.6243	0.6682	0.7168	0.715	0.743	0.7374	0.7636
Generations	60	45.5	46.1	57.4	44.7	51.5	45.4	46.7	40.6
T. time (s)	48.1	58	69.1	95.1	84.9	103.3	96.6	104.4	94.7
Ending CR	0.5822	0.6477	0.6766	0.7308	0.7383	0.7589	0.7664	0.7729	0.7785
Test CR	0.4804	0.5542	0.5495	0.5028	0.5215	0.4897	0.4776	0.486	0.472
OIGA_Asce	Add Att. 5	Add Att. 9	Add Att. 6	Add Att. 7	Add Att. 2	Add Att. 3	Add Att. 1	Add Att. 8	Add Att. 4
Initial CR	0.3533	0.3981	0.4308	0.4402	0.5112	0.5383	0.6215	0.6682	0.7112
Generations	49.5	49.2	46.9	53.3	49.3	59.5	57.6	55.4	46.2
T. time (s)	40.2	64.1	70.7	90.8	92.9	118.3	126.7	132.6	119
Ending CR	0.3963	0.428	0.4607	0.5243	0.5533	0.629	0.7065	0.7355	0.7607
Test CR	0.3252	0.3411	0.3458	0.3505	0.3579	0.3963	0.4	0.4215	0.4299
OIGA_Orig	Add Att. 1	Add Att. 2	Add Att. 3	Add Att. 4	Add Att. 5	Add Att. 6	Add Att. 7	Add Att. 8	Add Att. 9
Initial CR	0.3467	0.5374	0.571	0.6607	0.7103	0.6991	0.7449	0.7056	0.7308
Generations	57.7	59.2	59.8	55	57.6	56.9	47.8	52.9	49.4
T. time (s)	47.5	74.8	91.3	95.5	108.2	113.1	105.8	121.9	116.5
Ending CR	0.5477	0.5981	0.6869	0.7299	0.7467	0.7645	0.7692	0.7804	0.7879
Test CR	0.371	0.4196	0.4439	0.4598	0.4579	0.4505	0.4598	0.4458	0.4458
OIGA_Rand	Add Att. 8	Add Att. 3	Add Att. 5	Add Att. 7	Add Att. 4	Add Att. 2	Add Att. 6	Add Att. 9	Add Att. 1
Initial CR	0.3196	0.5243	0.5757	0.6121	0.6364	0.6766	0.6748	0.6729	0.6972
Generations	48.6	54.6	46.2	45.2	47.7	42.2	43.2	43.7	46.8
T. time (s)	35.9	65.5	64.6	70.6	81	76.9	83	89.9	102.9
Ending CR	0.472	0.5953	0.6159	0.6477	0.6879	0.6935	0.7028	0.7093	0.7449
Test CR	0.4654	0.4449	0.415	0.3944	0.4262	0.4346	0.4243	0.4215	0.4355

Summary	Normal GA	OIGA_Desc	OIGA_Asce	OIGA_Orig	OIGA_Rand
Initial CR	0.3308	0.3579	0.3533	0.3467	0.3196
Generations	167.3	437.9	466.9	496.3	418.2
T. time (s)	354.6	754.2	855.3	874.6	670.3
Ending CR	0.5897	0.7785 (31.6%)	0.7607 (29.0%)	0.7879 (33.6%)	0.7449 (26.3%)
Test CR	0.3953	0.472 (19.4%)	0.4299 (8.8%)	0.4458 (12.8%)	0.4355 (10.2%)

mutationRate = 0.01, crossoverRate = 1, survivorsPercent = 50%, ruleNumber = 30, popSize = 200, stagnationLimit = 30.
For each stage of OIGAs, generationLimit = 60; for normal GA, generationLimit = 540.

for incremental training. Therefore, classifiers can evolve their solutions incrementally in any situation where the introduction of new attributes may be frequent. When the available attributes are known, the individual discriminating ability of these attributes can be determined, and the order of attributes for incremental training can be selected accordingly. With the evaluation of classifier performance on four benchmark data sets and comparison to normal GAs, we find the use of OIGAs (especially an OIGA in descending order) can improve the performance of classifiers in terms of both training and test CRs. If we regard the whole incremental learning process as a complete training process, it also means continuous incremental training is a better way of training than batch-mode training.

The experimental results show that OIGA-Desc generally outperforms the other approaches in both training CR and test CR. This tells us that the order of

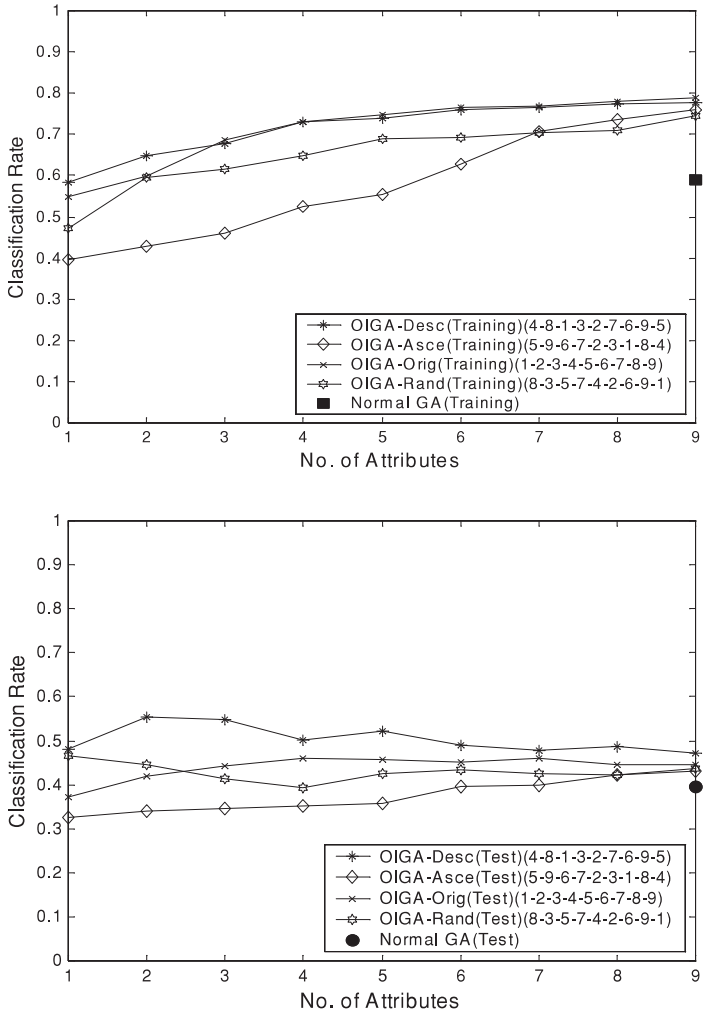


Figure 7. Comparison of OIGAs and normal GA on the glass data.

attributes presented for incremental training does affect the classifier performance. Introducing an attribute with higher discriminating ability in an earlier stage will improve the final performance. As a GA is a global searching algorithm that gradually improves its solutions, an early introduction of more contributing attributes will direct the candidate solutions to the potential final solution sooner and thus is more likely to converge to a better solution.

For some problems such as the yeast and glass data, the improvement versus a normal GA is significant. For problems such as the wine and cancer data, the improvement is relatively smaller. The results may be explained by the final accuracy achieved with these data, as we find the final accuracy for the wine and cancer

Table V. Performance comparison on the wine data—OIGAs and normal GA.

Summary	Normal GA	OIGA_Desc	OIGA_Asce	OIGA_Orig	OIGA_Rand
Initial CR	0.4034	0.5652	0.4517	0.5843	0.4416
Generations	144.7	289.3	429.6	446.7	412.1
T. time (s)	356.7	478.7	786.6	787.6	716.4
Ending CR	0.9876	0.9989 (1.1%)	0.9978 (1.0%)	0.9955 (0.8%)	0.9978 (1.0%)
Test CR	0.8303	0.9202 (10.8%)	0.8213 (−1.1%)	0.8607 (3.7%)	0.8281 (−0.3%)

mutationRate = 0.01, crossoverRate = 1, survivorsPercent = 50%, ruleNumber = 30, popSize = 200, stagnationLimit = 30.
For each stage of OIGAs, generationLimit = 40; for normal GA, generationLimit = 520.

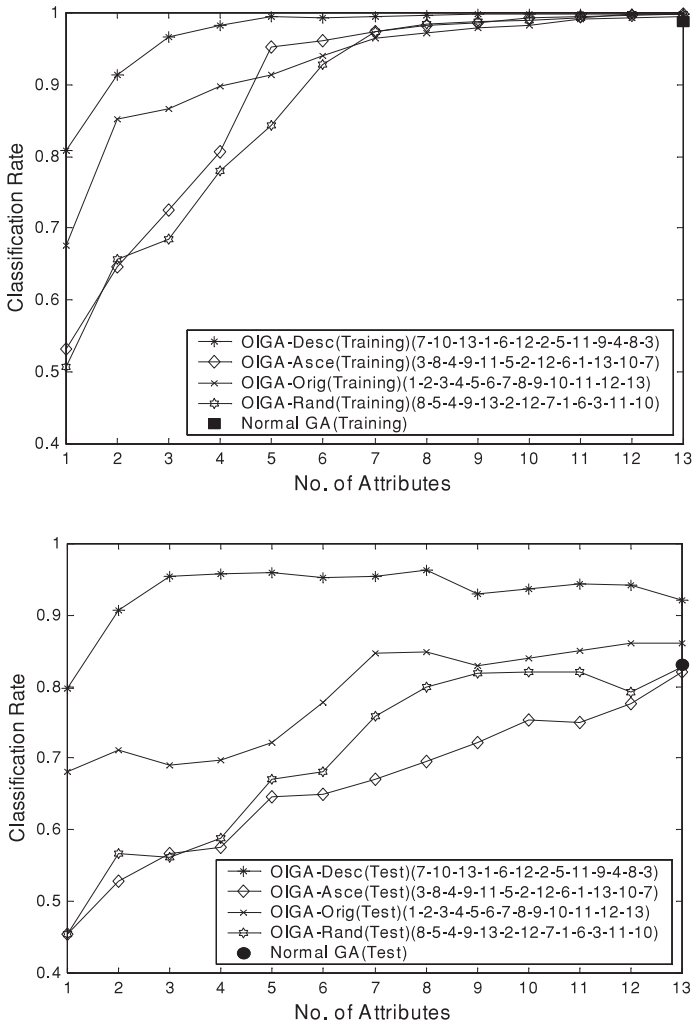


Figure 8. Comparison of OIGAs normal GA on the wine data.

Table VI. Performance comparison on the cancer data—OIGAs and normal GA.

Summary	Normal GA	OIGA_Desc	OIGA_Asce	OIGA_Orig	OIGA_Rand
Initial CR	0.7324	0.8997	0.7943	0.8324	0.835
Generations	136.6	505.8	520.6	403.7	524.6
T. time (s)	381.2	1177.3	1193.9	1007.8	1317.4
Ending CR	0.967	0.9673 (0.03%)	0.9713 (0.4%)	0.9696 (0.3%)	0.9708 (0.4%)
Test CR	0.9619	0.9668 (0.5%)	0.9479 (−1.5%)	0.9699 (0.8%)	0.9633 (1.5%)

mutationRate = 0.01, crossoverRate = 1, survivorsPercent = 50%, ruleNumber = 30, popSize = 100, stagnationLimit = 30.
For each stage of OIGAs, generationLimit = 60; for normal GA, generationLimit = 540.

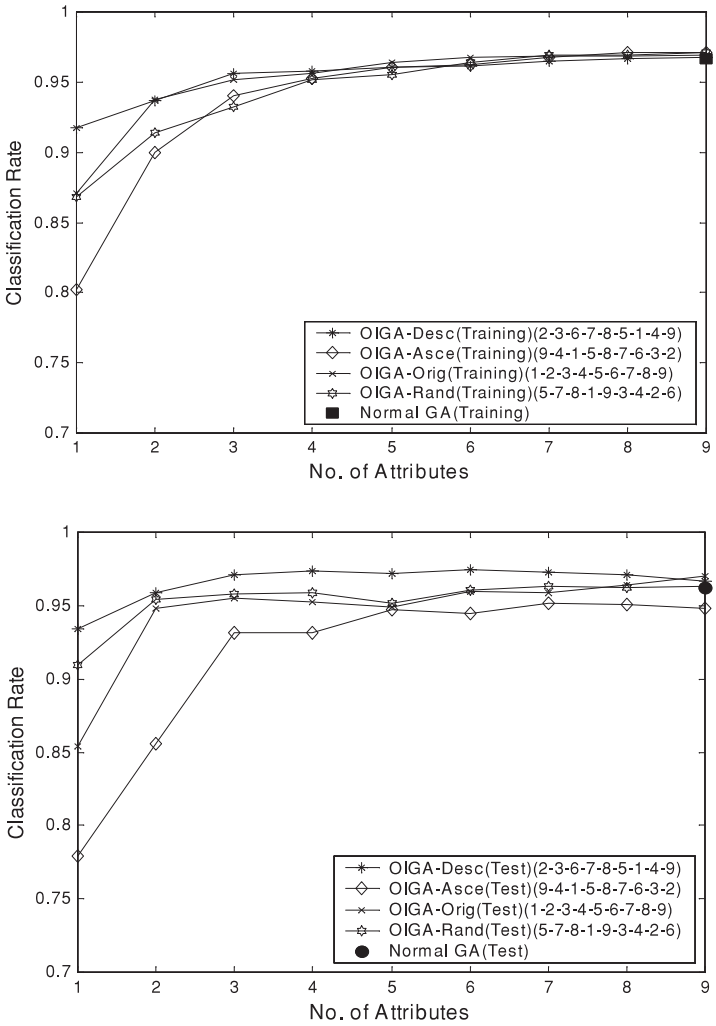


Figure 9. Comparison of OIGAs and normal GA on the cancer data.

data is relatively higher. That means these data are easier to classify, and therefore OIGAs may not demonstrate their advantage against the normal GA on these data.

There are some trade-offs and possible improvement for our design and experiments. First, as mentioned earlier, we have used the average of training and test CRs as the metric for measuring the individual discriminating ability of each attribute. There can be other options under different circumstances. For example, if a classifier is used for purely learning instances or the generalization of classifier is the only objective, the individual training CR or test CR may be used as the metric, respectively. Second, the incremental training method also provides another chance for attribute selection. It is found from the experimental results that the training and test CRs will be affected after the classifier learns a new attribute. They may drop a little as the new attribute is harmful to classification or increase as it facilitates classification. As a result, the training and test CRs are fluctuating during the whole training process. This fluctuation provides an opportunity to decide whether the new incoming attribute is selected into the final attribute set. For example, if the drop on the test CR exceeds a threshold after integrating a new attribute, it may signal that the introduction of this attribute will be harmful and thus it can be discarded.

6. CONCLUSIONS

This article proposes ordered incremental genetic algorithms (OIGAs) to address the continuous incremental training tasks of GA-based classifiers. Rather than learning input attributes in batch as done by normal GAs, OIGAs learn input attributes one after another. The classification rule sets are also expanded and evolved incrementally to accommodate the new attributes. As a preparatory step, attributes are arranged in different sequences by evaluating their individual discriminating ability, before they are presented in the training process.

By experimenting with different attribute orders, different approaches of OIGAs were evaluated on four benchmark data sets. The performance of each approach was compared with the others, and also with normal GAs. The simulation results showed that OIGAs can be successfully used for incremental training of classifiers and can achieve generally better performance than normal GAs. The order of attributes does have an effect on the final classifier performance, and OIGA training with a descending order of attributes has the best performance.

This article explores incremental training in a single classifier. Classifiers could also be implemented in a multi-agent environment, where classifier agents could exchange information on new attributes and classes. If available, they could also exchange evolved rule sets. They could help provide new training/test data, or even challenge each other with unsolved instances.

Acknowledgment

The first author is grateful to the Singapore Millennium Foundation for the scholarship awarded.

References

1. Anand R, Mehrotra K, Mohan CK, Ranka S. Efficient classification for multiclass problems using modular neural networks. IEEE Trans. Neural Netw 1995;6:117–124.
2. Guan SU, Li SC. Incremental learning with respect to new incoming input attributes. Neural Process Lett 2001;14:241–260.
3. Su L, Guan SU, Yeo YC. Incremental self-growing neural networks with the changing environment. J Intell Syst 2001;11:43–74.
4. Corcoran AL, Sen S. Using real-valued genetic algorithm to evolve rule sets for classification. In: Proc 1st IEEE Conf on Evolutionary Computation, Orlando; 1994. pp 120–124.
5. Guan SU, Ngoo CS, Zhu F. An intelligent product-brokering agent for m-commerce applications with user preference tracking. Electron Commerce Res Appl 2002;1:314–330.
6. Ishibuchi H, Nakashima T, Murata T. Performance evaluation of fuzzy classifier systems for multidimensional pattern classification problems. IEEE Trans Syst Man Cyber 1999; 29:601–618.
7. Setnes M, Roubos H. GA-fuzzy modeling and classification: Complexity and performance. IEEE Trans Fuzzy Syst 2000;8:509–522.
8. Merelo JJ, Prieto A, Moran F. Optimization of classifiers using genetic algorithms. In: Patel M, Honavar V, Balakrishnan K, editors. Advances in the Evolutionary Synthesis of Intelligent Agents. Cambridge, MA: MIT Press; 2001. pp 91–108.
9. Lanzi PL, Stolzmann W, Wilson SW. Learning classifier systems: From foundations to applications. Berlin: Springer; 2000.
10. Yamauchi K, Yamaguchi N, Ishii N. Incremental learning methods with retrieving of interfered patterns. IEEE Trans Neural Netw 1999;10:1351–1365.
11. Fu L, Hsu H, Principe JC. Incremental backpropagation learning networks. IEEE Trans Neural Netw 1996;7:757–761.
12. Polikar R, Udpa L, Udpa SS, Honavar V. Learn++: An incremental learning algorithm for supervised neural networks. IEEE Trans Syst Man Cyber 2001;31:497–508.
13. Guan SU, Zhu F. Learning of classifier agents based on incremental genetic algorithms. In: Proc 2002 IASTED International Conf on AI and Applications, Málaga, Spain, 2002. pp 353–357.
14. Guan SU, Zhu F. Incremental learning of collaborative classifier agents with new class acquisition: An incremental genetic algorithm approach. Int J Intell Syst 2003;18:1173–1193.
15. Michalewicz Z. Genetic algorithms + data structures = evolution programs, 3rd ed. New York: Springer; 1996.
16. Blake CL, Merz CJ. UCI Repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science, 1998.