An Organizational Coevolutionary Algorithm for Classification

Licheng Jiao, *Senior Member, IEEE* Jing Liu[1] Weicai Zhong

Institute of Intelligent Information Processing, Xidian University, Xi'an 710071, China

Abstract—Taking inspiration from the interacting process among organizations in human societies, a new classification algorithm, organizational coevolutionary algorithm for classification (OCEC), is proposed with the intrinsic properties of classification in mind. The main difference between OCEC and the available classification approaches based on evolutionary algorithms (EAs) is its use of a bottom-up search mechanism. OCEC causes the evolution of sets of examples, and at the end of the evolutionary process, extracts rules from these sets. These sets of examples form organizations. Because organizations are different from the individuals in traditional EAs, three evolutionary operators and a selection mechanism are devised for realizing the evolutionary operations performed on organizations. This method can avoid generating meaningless rules during the evolutionary process. An evolutionary method is also devised for determining the significance of each attribute, on the basis of which, the fitness function for organizations is defined. In experiments, the effectiveness of OCEC is first evaluated by multiplexer problems. Then OCEC is compared with several well-known classification algorithms on 12 benchmarks from the UCI repository datasets and multiplexer problems. Moreover, OCEC is applied to a practical case, radar target recognition problems. All results show that OCEC achieves a higher predictive accuracy and a lower computational cost. Finally, the scalability of OCEC is studied on synthetic datasets. The number of training examples increases from 100 000 to 10 million, and the number of attributes increases from 9 to 400. The results show that OCEC obtains a good scalability.

Index Terms—Data mining, classification, organization, evolutionary algorithms, coevolution.

[1] Corresponding author, e-mail: neouma@163.com, telephone: 86-029-88202661

I. INTRODUCTION

Evolutionary algorithms (EAs) [1], based on an analogy to natural evolution, have recently gained increasing interest. They are suitable for solving complex or ill-defined problems and have been successfully applied to the fields of numerical optimization, combinatorial optimization, machine learning, neural networks, and many other engineering problems [2]-[7]. Classification is one of the fundamental tasks of data mining [8]-[13] and can be described as follows [14]. The input data, also called the training set, consist of examples, each example having several attributes. Additionally, each example is tagged with a special class name. The objective of classification is to analyze the input data and to develop an accurate description for each class using the attributes present in the data. The class descriptions are used to classify future test data for which the class names are unknown. Applications of classification include credit approval, medical diagnosis, store location, etc. This paper introduces a new evolutionary algorithm for classification.

A. Related work

Classification has been studied extensively and the application of EAs to this field was initiated in the 1980s [15], [16]. Holland [15] and Smith [16] proposed two basic reference approaches, namely, the Michigan and Pittsburgh approaches, respectively. The Michigan approach maintains a population of individual rules which compete with each other for space and priority in the population. In contrast, the Pittsburgh approach maintains a population of variable-length rule sets which compete with each other with respect to performance on the domain task.

Neither of the approaches is perfect. The Michigan approach, which converges more rapidly, fails to learn good solutions for complex problems, whereas the Pittsburgh approach solves more difficult problems at a relatively high computational cost. Present, it seems that both approaches have their problems and advantages. This has prompted many researchers to develop new approaches along each one or hybrid ones, by selecting good features from both approaches and by avoiding the difficulties.

Choenni *et al.* in [17]-[20] proposed some algorithms based on the Michigan approach. They have made many improvements. They modified the individual encoding method to use nonbinary representation, and did not encode the consequents of rules into the individuals. Moreover, they used extended version of crossover and mutation operators suitable to their representations, did not allowing rules to be invoked as a result of the invocation of other rules, and defined fitness functions in terms of some measures of the classification performance.

De Jong *et al.* in [21] proposed an algorithm based on the Pittsburgh approach, GABIL. GABIL can continually learn and refine classification rules from its interaction with the environment. By incorporating a genetic algorithm (GA) as the underlying adaptive search mechanism, GABIL is able to construct a concept learning system that has a simple, unified architecture with several important features.

In order to alleviate the disadvantages of the Michigan and Pittsburgh approaches, some hybrid Michigan/Pittsburgh methodologies have been proposed, for example, COGIN [22], JoinGA [23], REGAL [24], and G-Net [25].

COGIN [22] is an inductive system based on GAs that exploits the conventions of induction from examples. Its novelty lies in the use of training set coverage to simultaneously promote competition in various classification niches within the model and constrain overall model complexity.

JoinGA [23] is a combination of the Michigan and Pittsburgh approaches together with a symbiotic niching component that can be used in multimodal classification. On the level of fixed-length individuals, JoinGA uses normal genetic operators. In order to be able to deal with multimodal concepts, JoinGA uses a new level of operators above the basic genetic level. The operators on this level put together and divide groups of individuals, building families out of single, cooperating individuals. In this way JoinGA keeps the Michigan type effective fixed-length individuals and corresponding simple crossover operations. JoinGA also avoids the problems of multiple solutions by using the Pittsburgh type families, so that the system may converge to a single highly fit family.

REGAL [24], a distributed GA-based system, designed for learning first order logic concept descriptions from examples. The population constitutes a redundant set of partial concept descriptions, and each evolves separately. G-Net [25] is a descendant of REGAL, and consistently achieves a better performance. The main features of the system include robustness with respect to parameter settings, use of the minimum description length criterion coupled with a stochastic search bias, use of coevolution as a high-level control strategy, the ability to face problems requiring structured representation languages, and the suitability to parallel implementation on a network of workstations.

Recently, many new approaches based on EAs for the classification task have been proposed. XCS [26], [27] acts as a reinforcement learning agent. It differs from traditional approaches in several respects [27]. First, XCS has a simplified structure since it does not have an internal message list. In addition, XCS uses a modification of Q-learning instead of the “bucket brigade” [15]. Most importantly, in XCS, classifier fitness is based on the accuracy of the classifier’s payoff prediction instead of the prediction itself. XCS represents a major development in learning classifier systems research, and has proved effective in many domains [28].

GEP [10] is a new approach for discovering classification rules by using genetic programming with linear representation. The antecedent of discovered rules may involve many different combinations of attributes. To guide the search process, [10] suggested a fitness function considering both the rule consistency gain and completeness. A multiclass classification problem was formulated as a combination of multiple two-class problems by using the one against all learning method. Compact rule sets were subsequently evolved using a two-phase pruning method based on the minimum description length principle.

DMEL [11] handles classification problems of which the accuracy of each prediction needs to be estimated. DMEL searches through the possible rule space using an evolutionary approach that has the following characteristics: 1) the evolutionary process begins with the generation of an initial set of simple, one-condition rules; 2) interestingness measure is used for identifying interesting rules; 3) fitness of a chromosome is defined in terms of the probability that the attribute values of a record can be correctly determined using the rules it encodes; and 4) the likelihood of predictions made is estimated so that subscribers can be ranked according to their likelihood to churn.

In economics, Coase in [29] explains the sizing and formation of organizations from the framework of transaction costs. This concept was introduced to the GA-based classifiers by

Wilcox in [30], which puts emphasis on inventing an autonomous mechanism using transaction costs for forming the appropriately sized organizations within a classifier.

There are many other approaches that also obtain good performances, such as EVOPROL [31], SIA [32], ESIA [33], EENCL [34], EPNet [35], etc. EAs are promising approaches for data mining, and have been applied to many problems other than the classification task. For example, Abutridy *et al.* in [12] presented a novel evolutionary model for knowledge discovery from texts, and Cano *et al.* in [13] carried out an empirical study of the performances of four representative EA models for data reduction in knowledge discovery in databases.

B. Proposed approach

Inspired by the idea of organizations [30], we propose a new evolutionary algorithm for classification, organizational coevolutionary algorithm for classification (OCEC). But the emphasis of OCEC is different from that of [30]. Since in the real world situation, organizations usually compete or cooperate with others so that they can gain more resources, OCEC does not put emphasis on forming the appropriately sized organizations, but on simulating the interacting process among organizations.

OCEC adopts the coevolutionary model of multiple populations, focusing on extracting rules from examples. It causes the evolution of sets of examples, and at the end of the evolutionary process, extracts rules from these sets. These sets of examples form organizations. Three evolutionary operators and a selection mechanism are devised to simulate the interaction among organizations. Additionally, because OCEC is inspired from the coevolutionary model, and considers the examples with identical class names as one population, it can handle multi-class learning in a natural way so that multiple classes can be learned simultaneously.

The main process of the existing EA-based and stochastic search classification algorithms [36], is first generating rules randomly, and then improving the quality of the rules by using the training examples. So these algorithms adopt an up-bottom search mechanism, and may generate meaningless rules during the evolutionary process. But OCEC adopts a completely different search mechanism. For classification, if the obtained description is represented as rules, then each rule covers some examples, and the examples covered by the same rule have some similarities in attribute values. Based on this, OCEC first clusters the examples with similar attribute values so as to form organizations, and then guides the evolutionary process by using the differing significance of attributes. At the end of the evolutionary process, rules are extracted from organizations. Such a process can avoid generating meaningless rules. Therefore, OCEC adopts a bottom-up search mechanism.

C. Organization of paper

In the remainder of this paper, OCEC is described in Section II. Section III evaluates the effectiveness of OCEC by multiplexer problems. Section IV compares OCEC with the available algorithms on benchmarks, and applies OCEC to a practical case, the radar target recognition problem. The scalability of OCEC is studied in Section V. Finally, conclusions and some ideas for the future work are presented in the last section.

II. AN ORGANIZATIONAL COEVOLUTIONARY ALGORITHM FOR CLASSIFICATION

A. Knowledge representation and relevant definitions

Present, since OCEC is devised to handle nominal data only, the continuous attributes are transformed into nominal ones by discretizing. For the sake of simplicity, each continuous attribute has been discretized by subdividing the range into 5 equal length intervals. In order to avoid confusion about terminology, some concepts are first introduced here.

Definition 1: Let $\overline{A_i}$ be a set of attribute values $\{a_{i1}, a_{i2}, \ldots, a_{ik}\}$. An instance space $\mathcal{I}$ is the Cartesian product of sets of attribute values, $\mathcal{I} = \overline{A_1} \times \overline{A_2} \times \ldots \times \overline{A_n}$. An attribute $A_i : \mathcal{I} \rightarrow \overline{A_i}$ is a projection function from the instance space to a set of attribute values. An instance i is an element of $\mathcal{I}$, and an example e is an element of $\mathcal{I}\times\mathcal{C}$, where $\mathcal{C}$ is a set of class names. The set of examples is labeled as $\mathcal{E}\subset\mathcal{I}\times\mathcal{C}$.

An attribute is a measurable feature of an instance. An instance is described by a vector of attribute values and an example is an instance together with a class name. The set of class names is fixed and is presented in advance. The examples are organized as a matrix, where each row represents an example and each column an attribute. Here is a simple example.

Table I The examples organized as a matrix

Name	Class	*SIZE*	*HAIR*	*EYES*
e_1	*A*	*short*	*golden*	*blue*
e_2	*A*	*tall*	*red*	*blue*
e_3	*A*	*tall*	*golden*	*blue*
e_4	*A*	*short*	*golden*	*gray*
e_5	*B*	*tall*	*golden*	*dark*
e_6	*B*	*short*	*dark*	*blue*
e_7	*B*	*tall*	*dark*	*blue*
e_8	*B*	*tall*	*dark*	*gray*
e_9	*B*	*short*	*golden*	*dark*

Example 1: Given a set of classified examples $\mathcal{E}=\{e_1, e_2, \ldots, e_9\}$. They are organized as a matrix shown in Table I. The instance space $\mathcal{I} = \overline{SIZE}\times\overline{HAIR}\times\overline{EYES}$, where $\overline{SIZE} = \{short, tall\}$, $\overline{HAIR} = \{golden, red, dark\}$, and $\overline{EYES} = \{blue, gray, dark\}$. The set of class names $\mathcal{C}=\{A, B\}$.

Definition 2: An **Organization**, *org*, is a set of examples with identical class names, and the intersection of different organizations is empty, that is,

$org \subseteq \mathcal{E}_c$, $c \in \mathcal{C}$, where $\mathcal{E}_c$ denotes the set of examples whose class names are c;

$\forall org_1, org_2 \subseteq \mathcal{E}_c, org_1 \neq org_2 \Rightarrow org_1 \cap org_2 = \varnothing$;

The examples in an organization are called **Members**.

Because each attribute has different significance in determining the class name of an instance, the attributes are classified into different types for an organization according to the information of the members, and thus is convenient for rule extraction at the end of the evolutionary process.

Definition 3: If all members of *org* have the same value for attribute A, then A is a **Fixed-value Attribute**. If A' is a fixed-value attribute and satisfies the conditions required for rule extraction, then A' is a **Useful Attribute**. The fixed-value attribute set of *org* is labeled as F_{org}, and the useful attribute set is labeled as U_{org}.

Because rules extracted from some organizations are meaningless, organizations are also classified into three types.

Normal organization: It is the organizations with more than one members and non-empty useful attribute set.

Trivial organization: It is the organizations with only one member. All attributes of such organizations are useful ones;

Abnormal organization: It is the organizations with empty useful attribute set.

The sets of the three types of organizations are labeled as ORG_N, ORG_T, and ORG_A, respectively. To satisfy the requirement of evolutionary operations, each organization need record some information. Therefore, an organization is represented as the following structure,

```
Organization = Record
    Member_List: Record the name of each member in this organization;
    Attribute_Type: Record the type of each attribute in this
    organization, that is, fixed-value attribute, useful attribute,
    or others;
    Organization_Type: Record the type of this organization, that
    is, trivial organization, abnormal organization, or normal
    organization;
    Member_Class: Record the class name of all members;
    Fitness: Record the fitness of this organization;
End.
```

B. Fitness function for organizations

After analyzing the relation between attributes and examples, we think that there are two factors to be considered in devising the fitness function for organizations,

(1) The number of members: The more members an organization has, the better the quality of the rule extracted from it. Therefore, the fitness of an organization should increase with the number of members.

(2) The number of useful attributes: Useful attributes will be used to generate rules at the end of the evolutionary process. The more useful attributes are, the more conditions the rules have. In fact, the more conditions a rule has, the fewer examples the rule covers. But over-generalization will result if the conditions are too few. Therefore, the fitness of an

organization should not monotonically increase or decrease with the number of useful attributes.

Because not all attributes, but only those with high significance are desired to appear in the final rules, a measure, **Attribute Significance**, is introduced.

Definition 4: **Attribute Significance** is the ability of an attribute in determining the class name for an instance. The attribute significance of A is labeled as S_A, and the value of S_A is determined during the evolutionary process.

As can be seen, S_A reflects the distribution of the values of A in each class. In the evolutionary process, the value of S_A is identical for all populations so that all populations can coevolve. When populations evolve, S_A evolves also. The value of S_A is updated when computing the fitness of an organization. The details are shown in Algorithm 1.

Algorithm 1 Attribute Significance

t denotes the generation of the evolutionary process. The number of attributes is m. N is a predefined parameter. org is the organization under consideration and $org \notin ORG_T$. A_j denotes the jth attribute in F_{org}.

```
begin
    if ($t$=0) then for $i$:=1 to $m$ do $S^0_{A_i} := 1.0$;
    Determining $F_{org}$;     $U_{org} := \emptyset$;
    for $j$:=1 to $|F_{org}|$ do
    begin
        Randomly selecting an organization $org'$ satisfying
        $org'$.Member_Class ≠ $org$.Member_Class;
        if ($A_j \in F_{org'}$) and (the value of $A_j$ in $F_{org'}$ is different from that
        of $A_j$ in $F_{org}$) then $U_{org} := U_{org} \cup A_j$
        else Reducing $S^t_{A_j}$ according to (1) (Case1);
    end;
    if ($U_{org} \neq \varnothing$) then
    begin
        Randomly selecting $N$ examples whose class names are different
        from $org$.Member_Class;
        if (the combination of the attribute values in $U_{org}$ does not
        appear in the $N$ examples) then
            Increasing the attribute significance of all attributes
            in $U_{org}$ according to (1) (Case2)
        else $U_{org} := \varnothing$;
    end;
end.
```

$$S_A^{t+1} = \begin{cases} 0.9S_A^t + 0.05, & \text{Case1}, \\ 0.9S_A^t + 0.2, & \text{Case2}. \end{cases} \tag{1}$$

The parameter N not only ensures that the rules extracted from organizations are consistent to some extent, but also makes the algorithm robust against noise. If N is set to a larger value, the rule is more consistent, but the algorithm is more sensitive to noise. Since the value of S_A is restricted to the range of [0.5, 2], it is set to 1.0 at the beginning, and updated during the evolutionary process. When the conditions of Case1 in (1) are satisfied, S_A should be punished:

$$S_A^{t+1} = S_A^t - \frac{S_A^t - 0.05}{10} = 0.9S_A^t + 0.05\,. \tag{2}$$

When the conditions of Case2 in (1) are satisfied, S_A should be awarded:

$$S_A^{t+1} = S_A^t + \frac{2 - S_A^t}{10} = 0.9S_A^t + 0.2\,. \tag{3}$$

The conditions of Case1 and Case2 are the ones required for rule extraction.

The idea of Algorithm 1 is encouraged by the following observations. If the values of *A* do not concentrate in the same class, *A* has low significance. But if a combination of several attribute values is unique in a certain class, these attributes together have high significance. An example is shown to evaluate whether Algorithm 1 can correctly determine attribute significance.

Example 2: Following Example 1, Fig.1 shows the evolutionary process of the attribute significance for the three attributes, where the *x*-coordinate stands for generations. As can be seen, after running 10 generations, the attribute significance can be differentiated completely. The significance of *HAIR* is the highest whereas that of *SIZE* is the lowest. This result agrees with that of decision trees, and illustrates the usefulness of Algorithm 1.

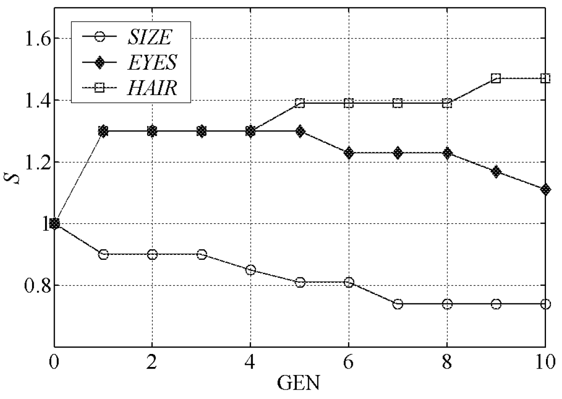

Fig.l. Evolutionary process of the attribute significance

On the basis of the attribute significance, the fitness function for organizations is defined as follows,

$$Fitness(org) = \begin{cases} 0, & org \in ORG_T, \\ -1, & org \in ORG_A, \\ |org| \prod_{i=1}^{|U_{org}|} S_{A_i}, & org \in ORG_N, \end{cases} \tag{4}$$

where A_i denotes the ith attribute in U_{org}. The fitness of trivial and abnormal organizations is set to 0 and –1, respectively, whereas that of normal organizations is the product of the number of members and each S_{A_i} in U_{org}. If $org \notin ORG_T$, the useful attributes must be determined by Algorithm 1 in advance.

C. Evolutionary operations for organizations

All evolutionary operations act on the members of organizations, and the traditional operators, such as crossover, mutation, and selection, cannot be used. Therefore, three new evolutionary operators and a selection mechanism are devised for organizations:

Migrating operator: First, two parent organizations, org_{p1} and org_{p2}, are randomly selected from a population. Next, n members, randomly selected from org_{p1}, are moved to org_{p2}, with two child organizations, org_{c1} and org_{c2}, obtained. Here $n \geq 1$.

Exchanging operator: First, two parent organizations, org_{p1} and org_{p2}, are randomly selected from a population. Next, n members, randomly selected from each parent organization, are exchanged, with two child organizations, org_{c1} and org_{c2}, obtained. Here $1 \leq n < \min\{|org_{p1}|, |org_{p2}|\}$, where $|org|$ denotes the number of members in org. The precondition for this operator is $|org_{p1}|>1$ or $|org_{p2}|>1$.

Merging operator: First, two parent organizations, org_{p1} and org_{p2}, are randomly selected from a population. Next, the members of the two organizations are merged, with one child organization, org_{c1}, obtained.

Organizational selection mechanism: The main idea of this mechanism is to encourage child organizations to compete with parent organizations. After an operator creates a pair of child organizations, a tournament is held between the child pair and the parent pair. The pair containing the organization with the highest fitness survives to the next generation, whereas the other pair is deleted. Since the three operators may generate abnormal organizations, and rules extracted from such organizations are meaningless, the mechanism must prevent such organizations from getting into the next generation. Therefore, when an abnormal organization survives to the next generation, it is dismissed and the members are added to the next generation as trivial organizations. Because the child organizations contain the same number of examples as the parent ones, the number of examples within a population remains constant. If only one organization remains in a population, it will be passed to the next generation directly.

D. Rule extraction from organizations and prediction method

For classification, one popular way of expressing the class descriptions is IF-THEN rules. Each rule has the form, **IF** <conditions> **THEN** <class name>. The <conditions> part (antecedent) of a rule contains a logical combination of attributes using the logical connective $\wedge$(AND) only, in the form, $term_1 \wedge term_2 \wedge \ldots \wedge term_n$. Each *term* is a triple <attribute, operator, value>, such as <*HAIR* = *golden*>. The <class name> part (consequent) of a rule contains the class name predicted for an instance whose attributes satisfy the <conditions> part.

When the evolutionary process is over, rules are extracted from organizations. In order to reduce the number of rules, all organizations are first merged by merging any two organizations in the same population into a new organization if the two organization satisfy: One useful attribute set is a subset of the other one and the values of the attributes in the intersection are identical. The members of the new organization are those of the two organizations, and the useful attribute set is the intersection of the two original sets, that is,

$$(U_{org_1} \subseteq U_{org_2}) \text{ or } (U_{org_2} \subseteq U_{org_1}) \Rightarrow (org = org_1 \bigcup org_2) \text{ and } (U_{org} = U_{org_1} \bigcap U_{org_2}). \qquad (5)$$

Next, a rule is extracted from each organization on the basis of the useful attribute set, i.e., each useful attribute forms a *term* in the <conditions> part, and the <class name> is equal to the Member_Class. For example,

Example 3: Following Example 1, if members of an organization are e_5 and e_9, and the useful attributes are *HAIR* and *EYES*, then the rule extracted from it is

IF (*HAIR* = *golden*) ∧ (*EYES* = *dark*) **THEN** (class name = *B*). □

In order to reduce the number of rules further, measures are taken as follows. Above all, a measure, **Relative Support**, labeled as *RS*, is calculated for each rule, which is derived from the ratio of positive examples a rule covers to all examples in the class the rule belongs to. On the basis of relative support, all rules are ranked. In order to prevent the rules of the classes with fewer examples from being positioned in tail, the rules are ranked based on the ratio, not on the number of positive examples. After all rules are ranked, some rules are deleted as follows. If the set of examples covered by a rule is a subset of the union of examples covered by the rules before this one, this rule is deleted. Algorithm 2 sums up the method for rule extraction.

Algorithm 2 Rules Extraction from Organizations

There are m populations, labeled as P_1, P_2, …, and P_m. E_r denotes the set of positive examples covered by rule r.

```
begin
```

```
    $RULES$:=$\emptyset$;
    for $i$:=1 to $m$ do
        while (there are two organizations in $P_i$, $org_1$ and $org_2$,
                satisfying the condition of (5)) do
        begin
            Merging $org_1$ and $org_2$ to form $org$ according to (5);
            $P_i := (P_i / \{org_1, org_2\}) \cup org$;
        end;
    Extracting a rule $r$ from each organization; Computing $RS_r$;
    $RULES := RULES \cup r$;
    Ranking the rules on the basis of the relative support;
    $i$:=1;
    while ($i \le |RULES|$) do
    begin
        if (there exist $k$ rules, $r_j$ ($j<i$, $j$=1, 2, …, $k$), satisfying
            $E_{r_i} \subseteq E_{r_1} \cup E_{r_2} \cup \ldots \cup E_{r_k}$) then Deleting $r_i$ from $RULES$;
        $i$:=$i$+1;
    end;
end.
```

Because rules extracted from different classes are put together and in most cases the rules for different classes may overlap, it is very important to adopt a suitable method to deal with conflicting rules and predict the class names of the instances.

Definition 5: Given that $i \in \mathcal{I}$ and $r \in RULES$. $|terms_r|$ denotes the number of terms in the <conditions> part of r, and $|terms_r^i|$ denotes the number of terms satisfied by instance i. The **Match Value**, MV_r^i between r and i is defined as $MV_r^i = |terms_r^i| / |terms_r|$.

According to the definition, the range of the match value is [0, 1], while 0 is the worst case and 1 the best case. The rule with the maximum match value is used to predict the class name for an instance. When more than one rules have the same maximum match value, the one ranked first is used.

E. Implementation of OCEC

Since a population is composed of the organizations with identical Member_Class, the number of populations is equal to the number of class names. The details of OCEC is presented in Algorithm 3.

```
Algorithm 3 Organizational Coevolutionary Algorithm for
            Classification
   There are $m$ class names, $c_1$, $c_2$, …, and $c_m$. The number of training
examples is $|example|$, and the $i$th example is labeled as $e_i$.
begin
   for $i$:=1 to $|example|$ do
      if (the class name of $e_i$ is $c_j$) then
         Add $e_i$ to population $P_j^0$ as a trivial organization;
   $t$:=0;
   while (the termination criteria are not reached) do
   begin
```

```
  $j$:=1;
  while ($j \le m$) do
  begin
     while (the number of organizations in $P_j^t$ > 1) do
     begin
        Randomly selecting two parent organizations, $org_{p1}$ and
        $org_{p2}$, from $P_j^t$;
        Randomly selecting an operator from the three
        evolutionary operators;
        Performing the selected operator on $org_{p1}$ and $org_{p2}$;
        Updating the attribute significance according to
        Algorithm 1 on the basis of $org_{c1}$ and $org_{c2}$;
        Computing the fitness of $org_{c1}$ and $org_{c2}$;
        Performing the selection mechanism on $org_{p1}$, $org_{p2}$ and
        $org_{c1}$, $org_{c2}$;
        Deleting $org_{p1}$, $org_{p2}$ from $P_j^t$;
     end;
     Moving the organization left in $P_j^t$ to $P_j^{t+1}$;     $j$:=$j$+1;
  end;
  $t$:=$t$+1;
end;
Extracting rules from all populations according to Algorithm 2.
```

end.

In order to have a full understanding of OCEC, an example is presented to show how multiple populations evolve and how rules are extracted from organizations.

Example 4: Following Example 1, n is set to 1. Because there are only 9 examples, parameter N is not fixed, but all examples of the other class are used. Fig.2 shows the organizations of each population at the 0th, 3rd, 6th, and 9th generation, respectively, with both the members and the useful attributes presented. At the beginning of the evolutionary process, populations A and B have 4 and 5 trivial organizations, respectively. During the evolutionary process, similar examples are clustered and new organizations are generated. Finally, populations A and B have 3 and 2 organizations, respectively. The rules extracted from organizations at the 9th generation are shown in Table II.

Table II The rules extracted from organizations at the 9th generation

	Organizations	**IF-THEN** rules
population A	org_1	**IF** (*HAIR* = *golden*) ∧ (*EYES* = *blue*) **THEN** (class name = *A*)
	org_2	**IF** (*HAIR* = *red*) ∧ (*EYES* = *blue*) ∧ (*SIZE* = *tall*) **THEN** (class name = *A*)
	org_4	**IF** (*HAIR* = *golden*) ∧ (*EYES* = *gray*) ∧ (*SIZE* = *short*) **THEN** (class name = *A*)
population B	org_1	**IF** (*HAIR* = *golden*) ∧ (*EYES* = *dark*) **THEN** (class name = *B*)
	org_2	**IF** (*HAIR* = *dark*) **THEN** (class name = *B*)

III. EVALUATION OF OCEC'S EFFECTIVENESS

The most important property of OCEC is evolving examples directly. Since the evolutionary operators and the fitness function for organizations just exhibit this property, this section conducts experiments to evaluate the effectiveness of the evolutionary operators and the fitness function by multiplexer problems.

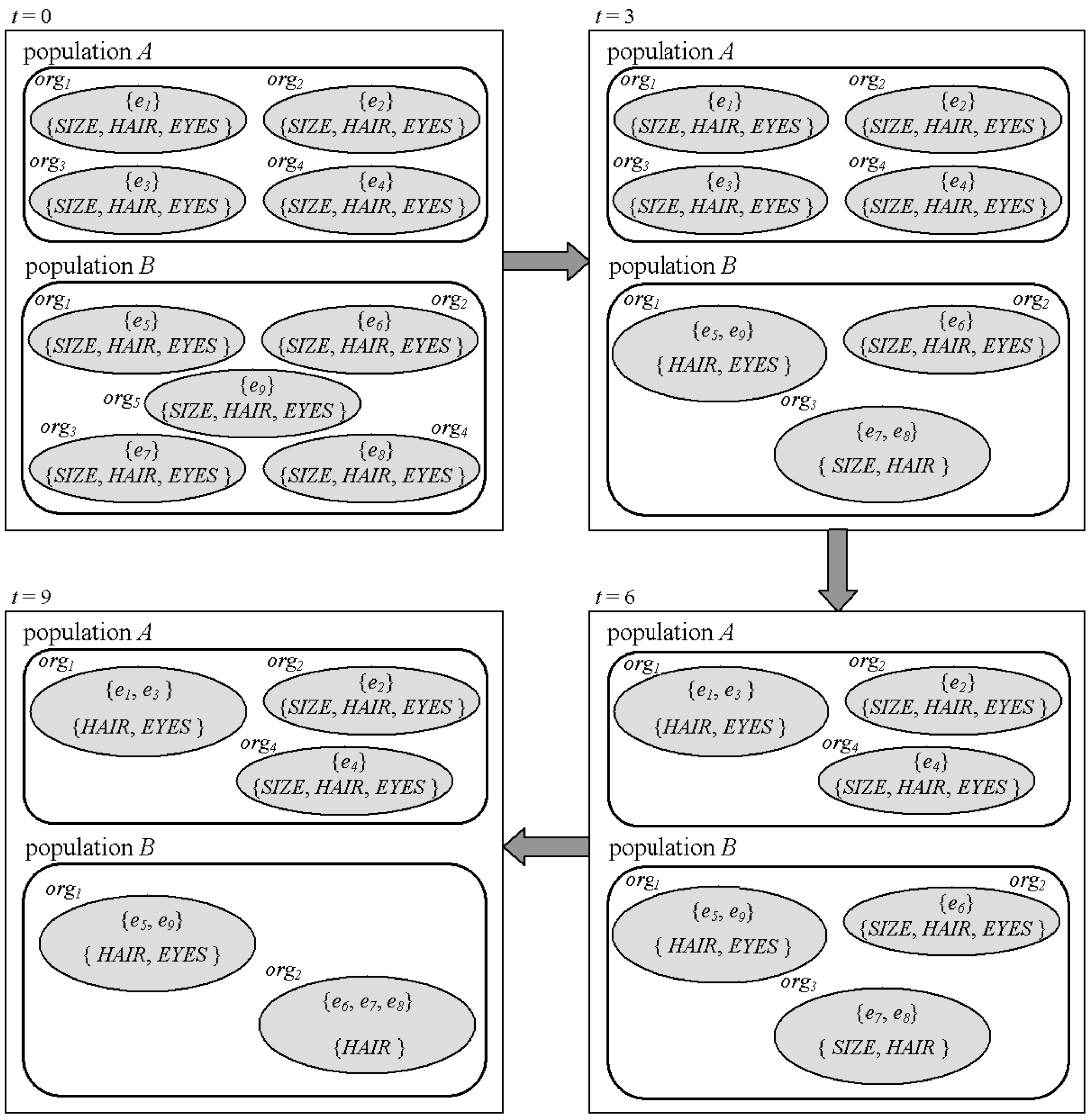

Fig.2. The evolutionary process of organizations

A. Multiplexer problems

Multiplexer problems were introduced to the machine learning community by Wilson in 1987 [37], and have often been used to evaluate the performance of learning classifier systems [26], [38]. Multiplexer problems are defined for strings of l bits, where $l=k+2^k$. The first k bits represent an address which indexes the remaining 2^k bits, and the function returns the value of the indexed bit. For instance, in the 6-multiplexer problem mp_6, we have that $\mathrm{mp}_6(100010)=1$, while $\mathrm{mp}_6(000111)=0$.

For an l-multiplexer problem, where $l=k+2^k$, the accurate and maximally general classifiers have (k+1) specific bits [26]. That is to say, there are (k+1) bits whose values are fixed, and the values of the other bits can be assigned arbitrarily. In the following experiments, each bit is represented as an attribute, and the value of the indexed bit is considered as the class name. Thus, the best IF-THEN rules have (k+1) terms.

B. Experimental results

The 20- and 37-multiplexer problems are used. The training set of the 20-multiplexer problem has 3000 examples, and that of the 37-multiplexer problem has 15 000 examples. The test set of each problem has 100 000 examples. Since the rule extraction is independent of the evolutionary process, to evaluate the effectiveness of the evolutionary operators and the fitness function, rules are extracted from the population at every 10 generations for the 20-multiplexer problem and every 100 generations for the 37-multiplexer problem, and used to predict the class names of the test examples. The parameter N is set to 10 percent of the number of the training set, and n is set to 1-5. For each multiplexer problem, 10 training and test sets are generated randomly. The evolutionary processes of the predictive accuracy, the number of rules, and the number of terms in each rule for the 10 independent runs are shown in Figs.3 and 4.

Figs.3 and 4 show that the predictive accuracies of all the 10 independent runs get higher along with the evolutionary process, and achieve 100% for both problems. In the meantime, the number of rules and terms in each rule decrease with the evolutionary process. For the 20-multiplexer problem, the number of terms in each rule reduces to 5 whereas for the 37-multiplexer problem it reduces to 6. These results accord with the characteristics of multiplexer problems introduced above, that is, for a (k+2^k)-multiplexer problem, the best IF-THEN rule has (k+1) terms.

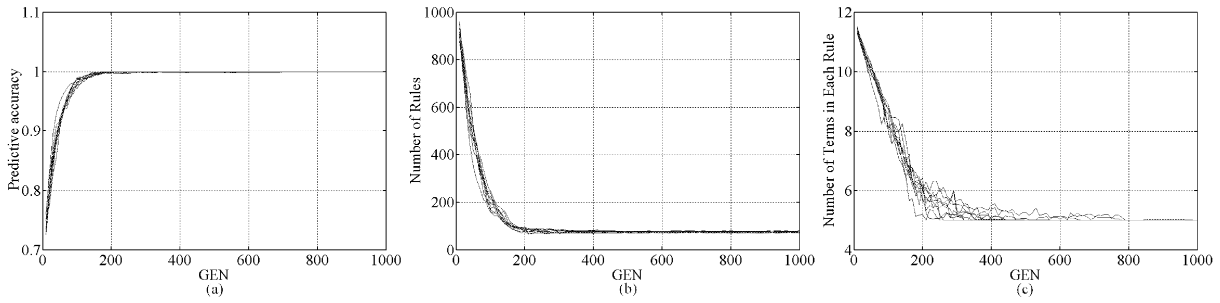

Fig.3. The evolutionary process of OCEC for the 20-multiplexer problem

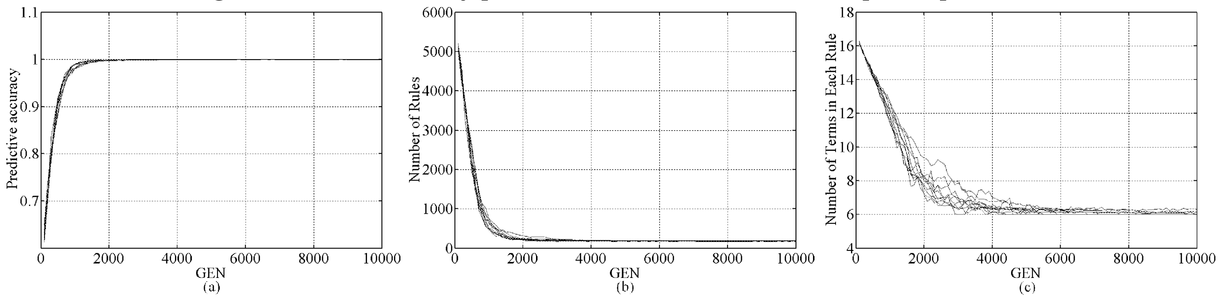

Fig.4. The evolutionary process of OCEC for the 37-multiplexer problem

The above experimental results illustrate that along with the effect of the evolutionary operators and the fitness function, OCEC not only evolves out the rules with high predictive accuracy, but also evolves out the accurate and maximally general rules.

IV. COMPARISON OF OCEC WITH AVAILABLE ALGORITHMS

A. Comparison on UCI repository datasets

In this section, 12 benchmarks from the UCI repository datasets [39] are used to test the performance of OCEC, and Table III shows the datasets and the test methods. For the Splice dataset, in order to be consistent with the compared algorithm, (2000+1190) is used in Section IV.A.1, and (2190+1000) is used in Section IV.A.2. In order to present a more stable estimate, 10-fold cross validation is used as the test method for small scale datasets. For large scale datasets, a training set is drawn randomly and the remainder is used as the test set. This process is repeated until ten training and test sets are generated.

The parameters of OCEC that need tuning are actually very few, and the following setting has been chosen. The number of generations is 500 for the datasets whose number of examples is less than 1000, and 1000 for the other datasets. *N* is set to 10 percent of the number of examples for each dataset, and *n* is fixed to 1 for all datasets. The results of OCEC on each dataset report the average predictive accuracy, the standard deviation, the average number of rules, and the average training time. All experiments are performed on a personal computer with Intel Pentium III 667 GHz processor as CPU, 128 MB of main memory.

Table III The UCI repository datasets used in experiments ("10-CV" represents the 10-fold cross validation, and "number + number" represents "the number of training examples + the number of test examples")

Datasets	#Examples	#Attributes	#Classes	Test Methods
Monk1	432	6	2	10-CV
Monk2	432	6	2	10-CV
Monk3	432	6	2	10-CV
Tictactoe	958	9	2	10-CV
Credit	690	15	2	10-CV
Breast cancer (W)	699	9	2	10-CV
Vote	435	16	2	10-CV
Australian	690	14	2	10-CV
Lymphography	148	18	4	10-CV
Mushrooms	8124	22	2	4000+4124
Chess (KR-vs-KP)	3196	36	2	2130+1066
Splice	3190	60	3	2000+1190 2190+1000

A.1 Comparison between OCEC and G-Net

G-Net was a new classification algorithm proposed in [25] and obtained a good performance. Here a comparison is made between OCEC and G-Net [25] on 9 different datasets of various sizes and difficulties. G-Net [25] used many other datasets, but only the 9 datasets were available at the UCI repository datasets. Due to the broad application of C4.5, its results are also given as a baseline. Table IV shows the comparison results and the best ones are shown in boldface. The performances of G-Net and C4.5 are those reported in [25].

As can be seen, the predictive accuracies of OCEC on 7 of the 9 datasets are equivalent to or higher than those of G-Net. Especially for the Tictactoe dataset, OCEC finds all the 8 possible ways to create a “three-in-a-row for x” and achieves 100% in predictive accuracy. The comparison on the Splice dataset is somewhat difficult since G-Net reports a classwise accuracy only. Therefore, a comparison in classwise accuracy between G-Net and OCEC is made, and the results of OCEC for all classes are also reported. The classwise accuracies of OCEC on the Splice dataset are slightly lower than those of G-Net. The number of rules obtained by OCEC is greater than that obtained by G-Net. This is probably due to the fact that only the logical connective AND is used in the IF-THEN rules. This makes each rule simple, but increases the size of the rule set. Because of the simplicity of each rule, this has little effect on the predicted efficiency.

A.2 Comparison between OCEC and JoinGA

JoinGA [23] is one of the best methods in GA-based classifiers. Here a comparison is made between OCEC and JoinGA on 5 different datasets. JoinGA [23] used many datasets, but only the 5 ones were available at the UCI repository datasets. The results of C4.5 are also used as a baseline. Table V shows the comparison results and the best ones are shown in boldface. The performances of JoinGA and C4.5 are those reported in [23].

As can be seen, the predictive accuracies of OCEC on 4 of the 5 datasets are equivalent to or higher than those of JoinGA. Only the predictive accuracy on the Splice dataset is little lower than those of JoinGA and C4.5. The training time for smaller datasets, such as the Australian and the Lymphography, are only 1.75s and 0.23s, respectively, and for larger datasets, such as the Chess and the Mushrooms, are 15.13s and 13.18s, respectively. The time for the Splice dataset is 179.56s since this dataset is more complex. The low computational cost of OCEC benefits mainly from the simple computations for the fitness function. Moreover, OCEC deals with multiple classes simultaneously by using the coevolutionary model, which also contributes to the low computational cost.

Table IV The comparison between OCEC and G-Net ("—" denotes that the item is not mentioned in the literature, and the results of OCEC are averaged over 10 runs.)

Datasets	Algorithms	Predictive accuracy (%)	Standard deviation (%)	Number of rules	Time (s)
Monk1	C4.5	**100.00**	**0.00**	—	—
	G-Net	**100.00**	**0.00**	**3.0**	—
	OCEC	**100.00**	**0.00**	11.0	0.22
Monk2	C4.5	67.17	10.66	—	—
	G-Net	**97.20**	**3.80**	**26.0**	—
	OCEC	73.18	7.31	28.1	1.18
Monk3	C4.5	**100.00**	**0.00**	—	—
	G-Net	**100.00**	**0.00**	**3.0**	—
	OCEC	**100.00**	**0.00**	6.0	0.15
Tictactoe	C4.5	92.93	1.82	—	—
	G-Net	99.03	0.62	10.5	—
	OCEC	**100.00**	**0.00**	**10.4**	0.67
Credit	C4.5	85.97	3.28	—	—
	G-Net	84.20	4.40	**14.0**	—
	OCEC	**87.97**	**4.38**	15.9	1.86
Breast cancer (W)	C4.5	94.15	3.32	—	—
	G-Net	94.71	2.89	**2.6**	—
	OCEC	**96.13**	**2.03**	17.2	1.41
Vote	C4.5	95.37	3.05	—	—
	G-Net	94.90	3.20	**2.0**	—
	OCEC	**95.87**	**2.61**	5.0	0.33
Mushrooms	C4.5	**100.00**	**0.00**	—	—
	G-Net	**100.00**	**0.00**	**3.0**	—
	OCEC	**100.00**	**0.00**	13.0	13.18
Splice	G-Net (EI)	**96.60**	—	7.0	—
	(IE)	**97.10**		10.0	
	(NE)	**96.70**		11.0	
	OCEC (EI)	95.98	—	—	—
	(IE)	94.98			
	(NE)	95.67			
	(All)	93.32	0.55	42.9	111.92

In the above, OCEC is compared with two GA-based classifiers on the UCI datasets. Since the UCI datasets have been widely used in testing the performances of various classifiers, high predictive accuracies have been obtained by many classifiers, such as EPNet [35], which is based on neural networks. Reference [35] reported that EPNet obtained high predictive accuracies on many UCI datasets, but we think EPNet is different from OCEC in intrinsic. EPNet does not explicitly express the uncovered patterns in a symbolic, easily understandable form, whereas OCEC uses a more understandable way, IF-THEN rules, to represent the results.

Table V The comparison between OCEC and JoinGA (The results of OCEC are averaged over 10 runs.)

Datasets	Algorithms	Predictive accuracy (%)	Standard deviation (%)	Number of rules	Time (s)
Australian	C4.5	87.0	3.1	—	—
	JoinGA	84.9	3.7	—	—
	OCEC	**87.97**	**4.04**	15.9	1.75
Lymphography	C4.5	79.8	8.4	—	—
	JoinGA	82.4	6.3	—	—
	OCEC	**86.38**	**8.92**	4.9	0.23
Chess (KR-vs-KP)	C4.5	99.5	—	—	—
	JoinGA	99.4	—	—	—
	OCEC	**99.51**	**0.09**	16.7	15.13
Mushrooms	C4.5	**100.0**	—	—	—
	JoinGA	**100.0**	—	—	—
	OCEC	**100.00**	**0.00**	13.0	13.18
Splice	C4.5	93.8	—	—	—
	JoinGA	**94.9**	—	—	—
	OCEC	93.34	0.52	45.5	179.56

B. Comparison of OCEC with XCS on multiplexer problems

XCS [26], [27] is one of the current state-of-the-art classifier systems, and [27] has made an in-depth research on the performance of XCS by multiplexer problems. This section presents a comparison between XCS and OCEC. Since XCS adopts an incremental mode, the experiments are designed as follows. 20- and 37- multiplexer problems are also used. The 10 training and test sets used in Section III.B are used for both OCEC and XCS. For OCEC, the number of generations is 1000 for the 20-multiplexer problem and 10 000 for the 37-multiplexer problem. For XCS, the training set is repeatedly used to train XCS, and finally the test set is used to check the predictive accuracy. The experiments for XCS are carried out with Butz's implementation [40] on the same computers as OCEC, and the parameters of XCS are set according to [27]. The comparison results averaged over 10 independent runs are shown in Table VI.

Table VI The comparison between OCEC and XCS (All results are averaged over 10 runs.)

Algorithms	20-multiplexer problem		37-multiplexer problem	
	Predictive accuracy (%)	Training time (s)	Predictive accuracy (%)	Training time (s)
OCEC	100	12.18	100	3610.99
XCS(a)	97.07	67.73	65.19	5394.40
XCS(b)	99.74	71.71	95.27	6707.96
XCS(c)	100	74.27	100	6914.69

In Table VI, for the 20-multiplexer problem, XCS(a), XCS(b), and XSC(c) stand for the training set has been repeatedly learned until 40 000, 50 000, and 60 000 examples are learned, and for the 37-multiplexer problem, they stand for 300 000, 400 000, and 500 000 examples are learned. As can be seen, although both the predictive accuracies of OCEC and XCS achieve to 100%, OCEC is much faster than XCS. In addition, XCS has a parameter, $P_{\#}$. $P_{\#}$ is used to control the number of terms in the rules, and has an important effect on the performance of XCS [27]. But the experimental results in Section III.B indicate, without any prior knowledge about multiplexer problems, OCEC can evolve out the accurate and maximally general rules automatically.

C. Radar target recognition problems

In this section, OCEC is applied to a practical case, radar target recognition problems. It refers to detecting and recognizing target signatures using high-resolution range profiles, for our case, in the inverse synthetic aperture radar (ISAR). A radar image represents a spatial distribution of microwave reflectivity sufficient to characterize the target illuminated. An important signature of the range profile is range resolution. It is related to the system bandwidth and represents the generally accepted measure of resolution of a range profile. Range resolution allows sorting the reflected signals on the basis of range. When range-gating or time-delay sorting is used to interrogate the entire range of the target space, a one-dimensional image, called a range profile, will result. Fig.5. is an example of such a signature for three different planes (B-52, Q-6, and Q-7) at 25°.

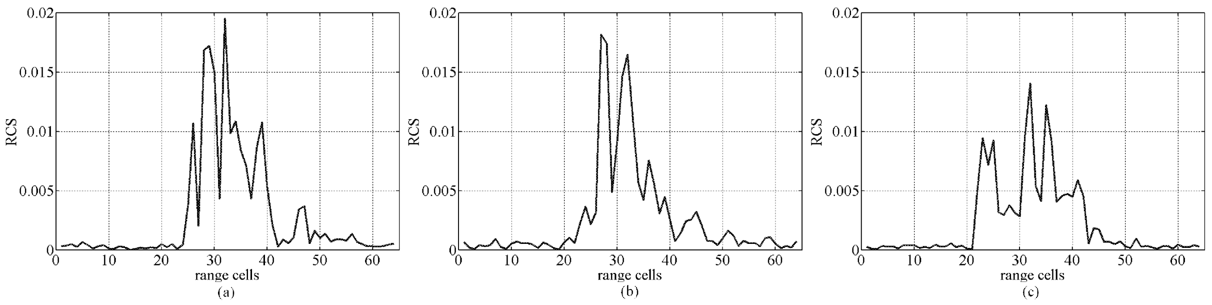

Fig.5. Range profiles of three different planes at 25°, (a) B-52, (b) Q-6, (c) Q-7

With the development of radar techniques, many methods based on imaging for the radar target recognition problem have been proposed, of which neural networks (NNs) [41] and support vector machines (SVMs) [42] are widely used. NNs have two drawbacks. One is that their architectures have to be determined a priori and the other is that NNs must map high-dimensional input spaces to low-dimensional input spaces. Although SVMs are independent of the dimension of the input space, it has to select the best kernel function. In this experiment, OCEC is used to classify the three planes, with higher predictive accuracies obtained.

The dataset used in this experiment is about three plane models, B-52, Q-6, and Q-7. It was acquired in a microwave anechoic chamber, and was composed of data in the angle range of 0°-150°. An example was obtained at every 0.5 degree, and total $301 \times 3 = 903$ examples were obtained. The dimension of each example is 64, and such a high input dimension will lead to poor results for many recognition techniques. To compare OCEC with the other methods, NNs and SVMs, we adopt the same sampling method, that is, the sampling of every other data is implemented for 150 training examples for every class, and the remaining examples are the test ones. The average predictive accuracies over 10 independent runs are shown in Table VII. The performances of NNs and SVMs are those reported in [41] and [42], respectively. As can be seen, OCEC outperforms the two other algorithms.

Table VII The experimental results for radar target recognition problems (Because [42] deals with two classes problem, there is no result in the cell of SVMs, B-52. All results are averaged over 10 runs.)

Planes	NNs (%)	SVMs (%)	OCEC (%)
B-52	90.00	—	**98.17±1.34**
Q-6	94.50	94.91	**96.79±1.82**
Q-7	86.00	94.41	**97.33±1.29**

V. SCALABILITY OF OCEC

The experimental results in the previous section show that OCEC achieves a good performance on small datasets. This section examines the scalability of OCEC along two dimensions, the number of training examples and the number of attributes. In the absence of a benchmark with large classification datasets, the evaluation methodology and synthetic datasets proposed in [8] are used. The parameters of OCEC are set as follows. The number of generations is 5000 for all datasets, and n is selected from 1-5 randomly. N is the same to that of the previous section.

A. Synthetic datasets

In [8], 10 classification functions were used to produce data distributions of varying complexities, where the functions F5 and F10 were the hardest to characterize and led to the highest classification errors. Moreover, the two functions have been used to examine the scalability of many algorithms [14]. Therefore, the following experiments are executed on the two functions.

Table VIII lists the attributes used in F5 and F10. There are two class names, A and B. We only specify the predicate function for A. All examples are not selected by the predicate function belong to B. F5 and F10 are defined as follows,

F5: A: ((**age** < 40) ∧
(((50K ≤ **salary** ≤ 100K) ? (100K ≤ **loan** ≤ 300K) : (200K ≤ **loan** ≤ 400K)))) ∨
((40 ≤ **age** < 60) ∧
(((75K ≤ **salary** ≤ 125K) ? (200K ≤ **loan** ≤ 400K) : (300K ≤ **loan** ≤ 500K)))) ∨
((**age** ≥ 60) ∧
(((25K ≤ **salary** ≤ 75K) ? (300K ≤ **loan** ≤ 500K) : (100K ≤ **loan** ≤ 300K))))

F10: **hyears** < 20 ⇒ **equity** = 0
hyears ≥ 20 ⇒ **equity** = 0.1 × **hvalue** × (**hyears** – 20)
disposable = (0.67 × (**salary** + **commission**) – 5000 × **elevel** + 0.2 × **equity** – 10K)
A: **disposable** > 0

Where ***P*** ? ***Q*** : ***R*** is equivalent to the sequential conditional function, i.e., the expression is equivalent to (***P*** ∧ ***Q***) ∨ (¬***P*** ∧ ***R***).

Table VIII Description of the attributes used in F5 and F10

Attribute	Description	Value
salary	salary	uniformly distributed from 20 000 to 150 000
commission	commission	if **salary** ≥ 75 000 ⇒ commission = 0 else uniformly distributed from 10 000 to 75 000
age	age	uniformly distributed from 20 to 80
elevel	education level	uniformly chosen from 0 to 4
car	make of the car	uniformly chosen from 1 to 20
zipcode	zip code of the town	uniformly chosen from 9 available zipcodes
hvalue	value of the house	uniformly distributed form 0.5*k*100 000 to 1.5*k*100 000 where *k*∈ {1, 2, …, 9} depends on **zipcode**
hyears	years house owned	uniformly distributed from 1 to 30
loan	total loan amount	uniformly distributed form 0 to 500 000

Since the attributes, **salary**, **commission**, **age**, **hvalue**, **hyears**, and **loan**, are continuous, they must be discretized previously. After descretizing, there are some examples whose attribute values are identical. Therefore, such examples are merged before they are used to train the algorithm, and the times used to merge such examples are also considered in the following experiments. In order to test the predictive accuracy of the obtained rules, another 10 000 instances are generated for each function as the test set.

B. Scalability on the number of training examples

Fig.6(a) shows the performance of OCEC as the number of training examples increases from 100 000 to 10 million in steps of 1 100 000. This corresponds to an increase in total database size from 4MB to 400MB.

The results show that OCEC has a linear classification time. Even when the number of training examples increases to 10 million, the classification time is still shorter than 3500 seconds. In addition, all the predictive accuracies of F5 range from 95.5% to 97%, and those of F10 range from 97.5% to 99.5%. The predictive accuracies of [8] on the two functions are only about 90%.

C. Scalability on the number of attributes

Since the original synthetic datasets have only 9 attributes, extra attributes are created by adding randomly generated values to each example. Note that the extra attributes do not substantially change the final rules because their attribute significance is very low. They simply increase the classification time. The number of training examples is fixed at 100 000. The number of attributes increases from 9 to 400 in steps of 39. This corresponds to an increase in the database size from 4MB to 160MB. Fig.6(b) shows the performance of OCEC.

The results show that OCEC still has a linear classification time. Even when the number of attributes increases to 400, the classification time is still shorter than 1400 seconds. In addition, because the values of extra attributes distribute uniformly in each class, their attribute significance is very low. Therefore, all predictive accuracies of F5 are still about 96%, and those of F10 are still about 98%.

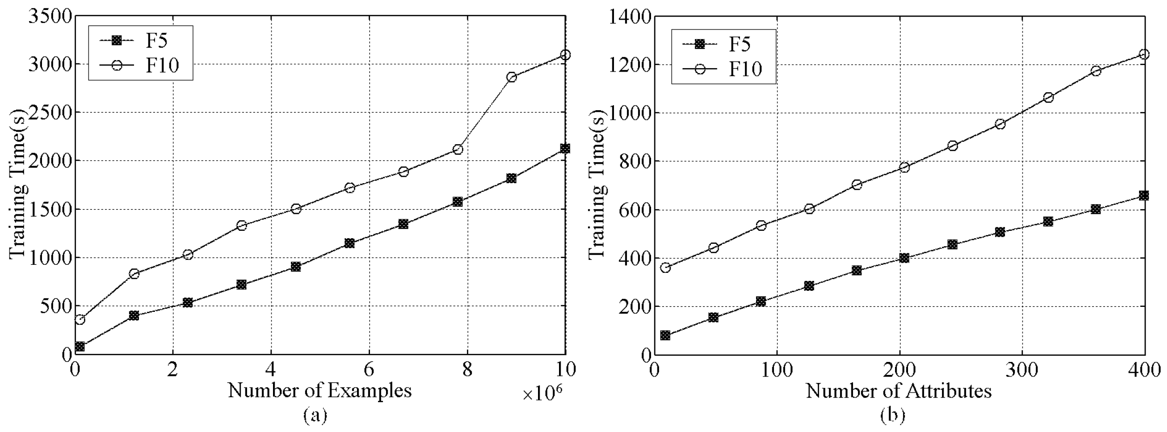

Fig.6 (a) The scalability of OCEC on the number of training examples, (b) the scalability of OCEC on the number of attributes

VI. CONCLUSION AND FUTURE WORK

Based on the interacting process among organizations in human societies, a new classification algorithm, OCEC, has been proposed in this paper. The results in Tables IV-VII show OCEC can learn the IF-THEN rules whose accuracy compares favorable to that achieved by some well-defined learners. In fact, the performance of OCEC is the best on most of the datasets we used. Additionally, OCEC has a low computational cost. All results of OCEC are obtained without performing any specific tuning, and it is proved that OCEC is quite robust and easy to use.

The good performance of OCEC benefits mainly from the bottom-up search mechanism, which enables OCEC to make full use of the information in examples. Furthermore, since the computations for the fitness function are simple, the computational cost of OCEC is very low.

Finally, the scalability of OCEC is studied, and the results show that OCEC achieves good scalability. Therefore, OCEC is an attractive tool for data mining.

The experimental results of Section IV.A.1 show that the number of rules obtained by OCEC is greater than that obtained by G-Net. This is probably due to the fact that only the logical connective AND is used in IF-THEN rules. Therefore, one of the future works is to use more logical connectives to reduce the number of rules. There are also other aspects of OCEC that need to be improved, such as updating the attribute significance in a better way, defining better fitness function, etc. The field of attribute selection has gained increasing interest in recent years [43]. OCEC could be also applied to this field in the future.

ACKNOWLEDGMENT

This research is supported by the National Natural Science Foundation of China under Grant 60133010. The authors are grateful to the reviewers and Professor Xin Yao for their helpful comments and valuable suggestions.

REFERENCES

[1] T. Bäck, U. Hammel, and H.-P. Schwefel, “Evolutionary computation: comments on the history and current state,” *IEEE Trans. Evol. Comput.*, 1(1), pp.3-17, 1997.

[2] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs,* 3rd Revised and Extended ed. New York: Springer-Verlag, 1996.

[3] D. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. Wiley-IEEE Press: New York, 2nd Edition, 1999.

[4] M. Mitchell, *An Introduction to Genetic Algorithms*. MIT Press: Cambridge, MA, Reprint Edition, 1998.

[5] T. Bäck, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press: Oxford, 1996.

[6] W. Zhong, J. Liu, M. Xue, and L. Jiao, “A multiagent genetic algorithm for global numerical optimization,” *IEEE Trans. Syst., Man, and Cybern. B*, 34(2), pp.1128-1141, 2004.

[7] W. Zhong, J. Liu, and L. Jiao, “A multiagent evolutionary algorithm for constraint satisfaction problems,” *IEEE Trans. Syst., Man, and Cybern. B,* to be published, 2005.

[8] R. Agrawal, T. Imielinski, and A. Swami, “Database mining: a performance perspective,” *IEEE Trans. Knowledge and Data Engineering*, 5(6), pp.914-925, 1993.

[9] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2000.

[10] C. Zhou, W. Xiao, T. M. Tirpak, and P. C. Nelson, “Evolving accurate and compact classification rules with gene expression programming,” *IEEE Trans. Evol. Comput.*, 7(6), pp.519-531, 2003.

[11] W. H. Au, K. C. C. Chan, and X. Yao, “A novel evolutionary data mining algorithm with applications to churn prediction,” *IEEE Trans. Evol. Comput.*, 7(6), pp.532-545, 2003.

[12] J. A. Abutridy, C. Mellish, and S. Aitken, “A semantically guided and domain-independent evolutionary model for knowledge discovery from texts,” *IEEE Trans. Evol. Comput.*, 7(6), pp.546-560, 2003.

[13] J. R. Cano, F. Herrera, and M. Lozano, “Using evolutionary algorithms as instance

selection for data reduction in KDD: an experimental study," *IEEE Trans. Evol. Comput.*, 7(6), pp.561-575, 2003.

[14] M. Mehta, R. Agrawal, and J. Rissanen, "SLIQ: a fast scalable classifier for data mining," in *Proc. 5th Int. Conf. Extending Database Technology*, Springer-Verlag, pp.18-23, 1996.

[15] J. H. Holland, "Escaping brittleness: the possibilities of general purpose learning algorithms applied to parallel rule-based systems," *Machine Learning: An AI Approach*, Los Altos, CA: Morgan Kaufmann, volume II, pp.593-623, 1986.

[16] S. F. Smith, "Flexible learning of problem solving heuristics through adaptive search," in *Proc. 8th Int. Joint Conf. Artificial Intelligence*, Karlsruhe, Germany: Morgan Kaufmann, pp.422-425, 1983.

[17] A. Choenni, "Design and implementation of a genetic-based algorithm for data mining," in *Proc. 26th Int. Conf. Very Large Data Bases*, Cairo, Egypt, pp.33-42, 2000.

[18] M. V. Fidelis, H. S. Lopes, and A. A. Freitas, "Discovering comprehensible classification rules with a genetic algorithm," in *Proc. 2000 Congress Evolutionary Computation*, San Diego, CA, pp.805-810, 2000.

[19] A. A. Freitas, "Understanding the critical role of attribute interaction in data mining," *Artif. Intell. Rev.*, vol. 16, pp.177-199, 2002.

[20] W. Kwedlo and M. Kretowski, "Discovery of decision rules from databases: an evolutionary approach," in *Proc. 2nd European Symp. Principles of Data Mining and Knowledge Discovery*, Nantes, France, pp.370-378, 1998.

[21] K. A. De Jong, W. Spears, and D. F. Gordon, "Using genetic algorithms for concept learning," *Machine Learning*, 13(2-3), pp.155-188, 1993.

[22] D. P. Greene and S. F. Smith, "Competition-based induction of decision models from examples," *Machine Learning*, 13(2-3), pp.229-257, 1993.

[23] J. Hekanaho, *An Evolutionary Approach to Concept Learning*. PhD thesis, Department of Computer Science, Abo Akademi University, 1998.

[24] A. Giordana and F. Neri, "Search-intensive concept induction," *Evol. Comput.*, 3(4), pp.375-416, 1995.

[25] C. Anglano and M. Botta, "NOW G-Net: learning classification programs on networks of workstations," *IEEE Trans. Evol. Comput.*, 6(5), pp.463-480, 2002.

[26] S. W. Wilson, "Classifier fitness based on accuracy," *Evol. Comput.*, 3(2), pp.149-175, 1995.

[27] M. V. Butz, T. Kovacs, P. L. Lanzi, and S. W. Wilson, "Toward a theory of generalization and learning in XCS," *IEEE Trans. Evol. Comput.*, 8(1), pp.28-46, 2004.

[28] P. L. Lanzi, W. Stolzmann, and S. W. Wilson, *Learning Classifier Systems. From Foundations to Applications*. Springer-Verlag: Berlin, LNAI vol. 1813, 2000.

[29] R. H. Coase, *The Firm, the Market, and the Law*. University of Chicago Press: Chicago, 1990.

[30] J. R. Wilcox, *Organizational Learning within a Learning Classifier System*. Master thesis, University of Illinois, Also Tech. Report No. 95003 IlliGAL, 1995.

[31] V. E. Castro, "Collaborative knowledge acquisition with a genetic algorithm," in *Proc. IEEE Int. Conf. Tools with Artificial Intelligence*, pp.270-277, 1997.

[32] G. Venturini, "SIA: a supervised inductive algorithm with genetic search for learning attributes based concepts," in *Proc. Eur. Conf. Machine Learning*, pp.280-296, 1993.

[33] J. J. Liu and J. T. Kwok, "An extended genetic rule induction algorithm," in *Proc. IEEE Congress on Evolutionary Computation*, pp.458-463, 2000.

[34] Y. Liu, X. Yao, and T. Higuchi, "Evolutionary ensembles with negative correlation learning," *IEEE Trans. Evol. Comput.*, 4(4), pp.380-387, 2000.

[35] X. Yao and Y. Liu, "A new evolutionary system for evolving artificial neural networks," *IEEE Trans. Neural Networks*, 8(3), pp.694-713, 1997.

[36] U. Rückert and S. Kramer, "Stochastic local search in k-term DNF learning," in *Proc. 20th Int. Conf. Machine Learning*, Washington DC, pp.648-655, 2003.

[37] S. W. Wilson, "Quasi-Darwinian learning in a classifier system," in *Proc. 4th Int. Workshop on Machine Learning*, Los Altos, CA: Morgan Kaufman Publishing, pp.59-65, 1987.

[38] S. W. Wilson, "Classifier systems and the animat problem," *Machine Learning*, vol. 2, pp.199-228, 1987.

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

[40] M. V. Butz, “An implementation of the XCS classifier system in C,” The Illinois Genetic Algorithms Laboratory, Univ. Illinois, Urbana-Champaign, IL, Tech. Rep. 99021, 1999.

[41] M. Wu, *Radar Target Identification Based on Computation Intelligence*. Master thesis, Xidian University, 1999.

[42] L. Zhang, W. Zhou, and L. Jiao, “Radar target recognition based on support vector machine,” in *Proc. World Computer Congress on Signal Processing*, Beijing, China, pp.1453-1456, 2000.

[43] I. Guyon and A. Elisseeff, “An introduction to variable and feature selection,” *Journal of Machine Learning Research*, 3, pp.1157-1182, 2003.

Licheng Jiao (SM'89) was born in Shaanxi, China, on Oct. 15, 1959. He received the B.S. degree from Shanghai Jiaotong University, Shanghai, China, in 1982. He received the M.S. and Ph.D. degrees from Xi'an Jiaotong University, Xi'an, China, in 1984 and 1990, respectively.

From 1984 to 1986, he was an Assistant Professor in Civil Aviation Institute of China, Tianjing, China. During 1990 and 1991, he was a Postdoctoral Fellow in the National Key Lab for Radar Signal Processing, Xidian University, Xi'an, China. Now he is the Dean of the electronic engineering school and the director of the Institute of Intelligent Information Processing of Xidian University. His current research interests include signal and image processing, nonlinear circuit and systems theory, learning theory and algorithms, optimization problems, wavelet theory, and data mining. He is the author of there books: *Theory of Neural Network Systems* (Xi'an, China: Xidian Univ. Press, 1990), *Theory and Application on Nonlinear Transformation Functions* (Xi'an, China: Xidian Univ. Press, 1992), and *Applications and Implementations of Neural Networks* (Xi'an, China: Xidian Univ. Press, 1996). He is the author or coauthor of more than 150 scientific papers.

Jing Liu was born in Xi'an, China, on Mar. 5, 1977. She received the B.S. degree in computer science and technology from Xidian University, Xi'an, China, in 2000, and received the Ph.D. degree in circuits and systems from the Institute of Intelligent Information Processing of Xidian University in 2004. Now she is a teacher in Xidian University.

Her research interests include evolutionary computation, multiagent systems, and data mining.

Weicai Zhong was born in Jiangxi, China, on Sept. 26, 1977. He received the B.S. degree in computer science and technology from Xidian University, Xi'an, China, in 2000, and received the Ph.D. degree in pattern recognition and intelligent systems from the Institute of Intelligent Information Processing of Xidian University in 2004. Now he is a postdoctoral fellow in Xidian University.

His research interests include evolutionary computation, multiagent systems, pattern recognition, and data mining.