11

Symbiotic Evolution of Rule Based Classifiers

Ramin Halavati[1] and Saeed Bagheri Shouraki[2]
[1]*Iranian Academic Centre for Education, Culture, & Research*
[2]*Sharif University of Technology*
Iran

1. Introduction

Genetic Algorithm is a widely used approach in predictive data mining where data mining output can be represented by If-Then rules and discovering the best rules is done by a genetic algorithm. The main motivation for using genetic algorithms in discovery of high-level prediction rules is that they perform a global search in the problem space and cope better with attribute interaction in compare with greedy rule induction algorithms often used in data mining (Freitas, 2001) and therefore, one can see the following papers for a wide variety of representation techniques and evolution approaches in this field: (Teng et al, 2004), (Hasanzadeh et al, 2004), (Chen & Linkens, 2004), & (Cordon et al, 1998) for evolution of weighted fuzzy rule base with simple linear genetic representation; (Golez & Dasgupta, 2002) for rule base evolution with binary tree representation; (Mendes et al, 2001) for a co-evolutionary approach which evolves fuzzy rules in one process and fuzzy membership functions in another process; (Ishibuchi & Yamamoto, 2004), (de la Iglesia et al, 2003), & (Lopes et al, 1999) use multi objective optimization approaches for rule base evolution; (Ishibuchi & Yamamoto, 2002) & (Tsang et al, 2005) for two stage evolution in which one stage generates candidate rules and the other stage selects a combination of them as a final rule base; (Riquelme et al, 2003) for hierarchical representation; and some other variations in (Zhu & Guan, 2004), (Goplan et al, 2006), (Gundo et al, 2004), & (Eggermont et al, 2003).
There are two basic strategies for rule base evolution task and many hybrid methods that combine the good features of these two methods. These basic approaches are Michigan approach exemplified by Holland's classifier system (Holland, 1986), and the Pittsburgh approach exemplified by Smith's LS-1 system (Smith, 1983). In this chapter, we will first study these two schools with more details in section 2 and show why there is a need for a third school, then introduce natural process of symbiogenesis in section 3 and symbiotic evolution as a novel solution for this approach in section 4. Then section 5 will present the experimental and comparison results, followed by the summary and concluding remarks in section 6.

2. Michigan and Pittsburgh schools for rule-based classifier evolution

There are two basic strategies for rule base evolution task and many hybrid methods that combine the good features of these two methods. These basic approaches are Michigan approach, introduced by John Holland (Holland, 1986), and the Pittsburgh approach, popularized by Ken De Jong and Steve Smith (Smith, 1983).

In Pittsburgh approach, a number of if-then rules are coded as a string and handled as an individual. The performance of each rule-set (i.e., each individual) is used as its fitness value. Thus the genetic search for finding rule-sets with high fitness values is equivalent to the search for rule-based systems with high performance. Hence, the optimization of rule-based systems is directly handled by genetic algorithms that try to maximize the fitness function. Some good rule-sets in a current population are inherited to the next population with no modification as elite individuals. The performance of each rule is not explicitly evaluated in Pittsburgh approach. Thus even if good rules exist in the current population, they are not always used for generating new rule-sets. Especially when good rules are included in poor rule-sets, they easily disappear during the generation update. Since a population consists of a number of rule-sets, long computation time and large memory storage are required in Pittsburgh approach (Ishibuchi et al, 1999). Interested reader can see (De Jong et al, 1993), (Janikow, 1993), (Sen et al, 1997), & (Smith 1983) as good examples of this approach.

On the other hand, in Michigan approach where a single if-then rule is coded as a string and handled as an individual, the performance of each rule is used as its fitness value. That is, the performance of rule-sets (the entire population of current rules) is not utilized in the genetic search for finding rule-based systems with high performance. Thus the optimization of rule-based systems is indirectly performed by searching for good if-then rules. Performance of the current rule-set is not explicitly evaluated in the genetic search of the Michigan approach. Thus a good rule-set can be destroyed by the generation update (i.e. the performance of the current population can be decreased). Since a population includes only a single rule-set, computation time and memory storage in Michigan approach are much smaller than those in Pittsburgh approach where a population consists of a number of rule-sets. In Michigan approach, good if-then rules in the current population (i.e., in the current rule-set) are inherited with no modification to the next population. The generation update in Michigan approach can be viewed as a partial change of the current population where bad rules are replaced with newly generated rules. Thus once good if-then rules are found, they are not likely to disappear. (Ishibuchi et al, 1999). To see some good examples, one can check (Holland, 1986) and (Wilson, 1987).

There are three main viewpoints from which Pittsburgh and Michigan approaches can be compared: First, Pittsburgh approach seems to be better suited at batch-mode learning (when all training instances are available before learning is initiate) and for static domains, and Michigan approach is more flexible to handle incremental-mode learning (training instances arrive over time) and dynamically changing domains (Corcoran & Sen, 1994).

Second, considering that many classifier systems need to cover a complex state space in a small group of cooperative rules, one will see that this is in contrast to the nature of Michigan approach in which the rules are intrinsically competitive and the Pittsburgh approach is more suited to the provision of cooperation. This is because the lack of competition between individual classifiers in the Pittsburgh method allows the algorithm to find novel cooperative solutions that the population-level GA can maintain and proliferate. Therefore, Pittsburgh approach is usually the method of choice to apply to problems that require the development of cooperative populations (Barry et al, 2004).

The third item is very similar to the second: As evolving rules of a Michigan process are rivals and the general fitness value of the population has no effect in evolution, two problems occur: First, we usually need strategies for detection and prevention of redundant

concept descriptions among population members (Liu & Kwok, 2000); Second, as a side effect of the first problem, a portion of training examples may be left unclassified and although the evolution would be at a stable position, there would be no rule for classification of this portion.

The fact that Pittsburgh is more powerful or easier to use for evolution of rule-sets in environments with complex concepts, where there is an urge for evolution of cooperative rules, makes it more attractive for most practical problems. However, the Pittsburgh approach presents its own limitation as well: In particular, because the evolution operates at a rule-set level, GA receives only high-level feedback from the fitness function and therefore cannot evaluate the role of individual rules in the success of a rule-set; hence, it requires a large additional effort to generate optimal populations. This increased effort in addition to the increased computational resource required to operate at the population level can present new challenges when devising efficient implementations for a Pittsburgh classifier evolution (Barry et al, 2004). This problem is a very important and known general problem of traditional genetic algorithms, called the linkage problem (Watson & Pollack, 2000).

Linkage problem has two parts: The first problem is called the problem of garbage or hitch-hiker genes (Forrest & Mitchel, 1993). In traditional GA, each chromosome may have a combination of good and bad genes which affect the total fitness value of the chromosome together. The effect of this problem in rule base evolution task is that a rule-set may have some rules with very good classification accuracy and some rules that have no positive effect or even have negative effects on the classification task. As evaluation is only done at rule-set level, selection or removal of all rules inside a rule-set is done together and there is no distinction between rules that have positive or negative effect on the classification. These bad rules (genes) inside a chromosome are called garbage genes or hitch-hiker genes because they gain their chance of survival by sticking to good genes as parastis.

The second part of Linkage problem is related to the recombination operator of genetic algorithms. During the process of this operator, some parts of the two parent chromosomes are extracted and merged with each other to create an offspring. Selection of appropriate parts from either of the parents has a great effect on the performance of the entire process, but there are many problem in which there is no way to identify the good sub-chromosomes. Here in rule-set evolution, one of the interesting features of the Pittsburgh approach is the evolution of cooperative rules inside a rule-set, but using a crossover operator separates the rules of one rule-set from each other and then blindly combines them with some from another rule-set, with no guarantee that these parts match each other or be able to help each other in a common classification task.

Many different recombination operators or alternative evolution strategies are introduced to cope with linkage problem in GA, such as designing more sophisticated recombination operators for simple genetic algorithms such as the ones with more number of cut points, random cut point positioning, uniform crossover, linear combination of genes, etc., see (Mitchell, 1999) for an extensive list; use of chromosome reordering operators and repositioning of genes inside the chromosome on the fly such as Inversion operator (Bagley, 1967) and Linkage Learning Genetic Algorithm (Harrik, 1997); and algorithms based on partially specified chromosomes such as Messy Genetic Algorithms (mGA) (Deb, 1991), (Goldberg et al, 1989), Cooperative Co-Evolutionary Algorithms (CCEA) (Potter & De Jong, 1994), Symbiotic Evolutionary Adaptation Model (SEAM) (Watson & Pollack, 2000), and Incremental Commitment Genetic Algorithm (ICGA) (Watson & Pollack, 1999).

As far as the authors know, except CCEA approach which is partially used in some tasks and some special purpose recombination operators, none of the other above approaches have been used in rule base evolution and the major efforts in rule-based classifier evolution to cope with linkage problem have been in hybridizations of Michigan and Pittsburgh approaches to add the positive features of both methods together, such as (Ishibuchi et al, 1999) & (Tan et al, 2003). Not commenting on the applicability or generality of these hybrid approaches, we present a novel pure approach based on Symbiotic evolution instead of Genetic evolution to solve this problem in the rest of this chapter. It must be emphasized that we introduce this algorithm as a basic approach comparable to pure Pittsburgh and therefore, it is not compared with hybrid approaches or extensions of other algorithms as all such hybridizations or extensions can be studied for this algorithm as well. Section 3 will represent the natural bases of this approach and section 4 will have all the details.

3. The natural process of symbiogenesis

The natural process of symbiogenesis (Merezhkovsky, 1909) is the creation of new species from the genetic integration of organisms, called symbionts. Symbiogenesis has enabled some of the major transitions in evolution (Maynard Smith & Szathmary, 1995), including the origin of eukaryotes which include all plants and animals. This kind of genetic integration is quite different from the transfer of genetic information in sexual reproduction. Sexual recombination occurs between similar organisms (i.e. of the same species) and involves the exchange of parts of the genome in a mutually exclusive manner; that is, every gene acquired from one parent is a gene that cannot be acquired from the other parent. In contrast, symbiotic combination may also occur between genetically unrelated organisms (i.e. different species) and involve the integration of whole genomes. The resultant composite may have all the genes from one symbiont and at the same time acquire any number of genes from the other symbiont (Watson & Pollack, 2000).
Based on this idea, symbiotic combination operator is introduced (Watson & Pollack, 1999) & (Watson & Pollack, 2000) as an alternative for sexual recombination operator. Symbiotic combination operator is applied to partially specified chromosomes, i.e., chromosomes which have some positions with unspecified values. This operator takes two partially specified chromosomes and makes an offspring with the aggregation of their characteristics of both of them; see Fig. 1 as an example. Therefore, in contrast to the standard crossover operator that receives two fully specified chromosomes and creates one/two individuals that have received each of their genes from either parents, this operator runs over two/more partially specified representations and creates an offspring with can have even all genes of both/all parents.

```
Chromosome A:        1--1---0
Chromosome B:        --00-111
A + B:               1-01-110
```

Fig 1. An example of symbiotic combination. Chromosomes A and B, each, have some unspecified locations, shown with '-' mark. Their combination has specified values for all locations that are specified in at least one of the donors. If there would be a conflict between the specified values, like the last gene of the above chromosomes, all conflicts are resolved in favor of one donor, here A.

This can be very beneficent for evolution of rule based classifiers in Pittsburg approach because each individual (chromosome) is a complete classifier. Therefore, its rules are a collection and they have proved to work good together. Separating them for a recombination and combining some parts of them with parts of another classifier may disrupt the functionality of both classifiers. On the other hand, adding them, assuming that each of them is a relatively good classifier, just adds up their classification powers.

4. Symbiotic evolutionary algorithm

The basic idea of Symbiotic Evolutionary Algorithm (SEA) is to replace the crossover operator of Pittsburgh genetic algorithm (PGA) with symbiotic combination operator. To do so, the evolution starts with rule-sets (individuals) which have just one rule (gene). During the process, similar to traditional PGA, evaluation and selection is done at rule-set level. Mutation operator is also quite similar to conventional PGA, but instead of crossover operator, sometimes two rule-sets combine using symbiotic combination and create an individual with more rules. If this combination shows a higher accuracy in compare to its parents, the parents are removed from the population and the offspring remains, otherwise, the offspring is neglected.
In this section, we first present our rule-set model which is used both in SEA and the PGA that is used in next section for comparisons. Then will move on the details of the Symbiotic Evolutionary Algorithm.

4.1 Rule-set model and fitness values

To emphasize on the algorithm, we have a chosen a very simplistic representation for our fuzzy rules, taken from (Hasanzadeh et al, 2004), but we still insist that SEA is not dependent to this model or the fuzzy nature of the rules. In this model, each rule is a horn clause, with If-part consisting of fuzzy membership functions for different features of the problem data base, and Then-part stating the class to which this rule belongs. A rule-set is composed of one or more rules, with each rule having a weight value stating its role in final decision. To classify an input by a rule-set, each of the rules computes the degree of similarity between the input and its own If-part and based on that, it states a degree of belief to its Then-part. Then, a weighted sum of the degree of beliefs for each class is computed and the class which gets the highest value is chosen. Fig. 2 specifies the structure of the rule-set.

```
<RULE-SET>  → a set of <RULE>s
<RULE>      → <WEIGHT> + a set of <CONDITION>s + <RESULT>
<WEIGHT>    → a real value
<RESULT>    → a Class Name
<CONDITION> → a <FEATURE> [IS / ISNOT] a <MEMBERSHIP FUCNTION>
<FEATURE>   → one of the features of dataset.
<MEMBERSHIP FUCNTION> →   one of the possible fuzzy values for the
                          respective feature.
```

Fig .2. Formal structure of the rule set (chromosome)

The fitness of each rule-set is defined as the accuracy of the rule-set in classification of all training data. Accuracy is a measure combining the classification soundness with 99.9

percent effect and the simplicity of the rules with 0.1 percent effect. The simplicity measure is used to break the tie between two rule-sets with different complexities and similar classification rate, in favor of the simpler rule-set. Simplicity of a rule-set is computed as stated in equation 1.

$$\text{Simplicity} = \frac{1 + \text{Number of rules with just one condition}}{\text{Total number of conditions in all rules}} \tag{1}$$

4.2 The algorithm

The Symbiotic Evolutionary Algorithm starts by generating a population of random rule-sets, each having just one rule. In each iteration of the algorithm, a set of rule-sets with high fitness values are selected using a tournament selection algorithm; they will be called the *Selected Set* hence forth. After selection, each of these individuals undergoes a mutation and all mutants are added to the population. The mutation operator is presented in Figure 3.

```
Function Name: MUTATATION

Summary:     Takes a rule set and mutates it.

Input: Rule Set R.
     Assume R ={R1,R2,...,Rn} and each Ri as
     [Weight+ (F1,C1,MF1) ∧(F2,C2,MF3)∧...∧(Fm,Cm,MFm), Class] where each
     Fj is feature, Oj is a condition(Is/Is Not), and MFj is a
     membership function from the domain of Fj.

Function Detail:
  1. Randomly choose Ri from R1 to Rn. Set m to the number of rules in
     Ri.
  2. Randomly select one of the next steps and apply it on Ri:
       a. Increase or decrease Weight.
       b. Choose j from 1..m, remove (Fj,Cj,MFj) from Ri.
       c. Randomly generate a new (F,C,MF) and concatanate it to Ri.
       d. Choose j from 1..m, reverse Cj so that Is becomes IsNot,
          and IsNot becomes Is.
       e. Choose j from 1..m, change MFj to a random new membership
          function from the domain of Fj.
  3. Return.
```

Fig. 3. Pseudo Code of the Mutation Operator

After mutation, instead of the conventional cross over operator, symbiotic combination operator is applied over the selected set. The operator takes two members of the selected rule-sets and merges them, so that the combination includes all rules of both sets. If the child strictly outperforms both of its parents, the combination will be added to the population; otherwise, it will be discarded. To control the growth speed of the number of rules in each rule-set, there is another control mechanism that limits the size of the largest rule-set that can be added to the population at a time. This parameter, which will be called *SizeLimit*, is 1 at the beginning and limits the size of rule-sets to just one rule. During the process, *SizeLimit* is increased with a selected strategy, and allows emergence of rule-sets with more number of rules. In all of our implementations, we have set the control strategy to a simple linear function of iterations count, but one may use a more complicated function, if it looks fit.

Fig. 4 presents the pseudo code of Symbiotic Evolutionary Algorithm.

```
Algorith Name: Symbiotic Evolutionary Algorithm

Summery:     Takes a database of training examples and generates a
             rule-set to classify them, using symbiotic combination
             operator and Mutation function.

Parameters:  SR:  Selection Rate
             TS:  Tournament Size
             RC:  Random Rule Creation Rate
             MP:  Maximum Population

Algorithm Detail:
  1. INITIALIZATION:
        a. Generate a population of random rule-sets, each having
           just one rule.
  2. PROCESS CONTROL:
        a. Update SizeLimit (Initialized to 1).
        b. If Best generated rule set is satisfactory, return it
           and exit.
  3. SELECTION PHASE:
        a. Create an empty set called SelectedSet.
        b. For SR x PopulationSize times,
              i. Randomly pick TS rule-sets from the pool, add the
                 best one to SelectedSet.
  4. MUTATION PHASE:
        a. For each memer of SelectedSet such as rs,
              i. Create a mutated copy of rs using Mutation
                 function, call it rs'.
             ii. Add rs' to the pool.
  5. SYMBIOTIC COMBINATION PHASE:
        a. For each two members of the SelectedSet such as rs1 and
           rs2,
              i. Create the symbiotic combination of rs1 and rs2
                 and call it rs3.
             ii. If SizeOf(rs3) < SizeLimit and fitness value of
                 rs3 exceeds that of rs1 and rs2,
                     Add rs3 to the pool.
  6. DIVERSITY MAINTANANCE:
        a. Create RC random new rule-sets and add them to the pool.
  7. POPULATION CONTROL:
        a. While PopulationSize is above MP limit, randomly select
           and remove some random rule-sets from the poool.
  8. Goto Step 2.
```

Fig. 4. Pseudo Code of Symbiotic Evolutionary Algorithm

5. Experimental results

5.1 Test conditions

There are too many classification approaches and also many extensions to basic genetic based classifiers. But as we are introducing SEA as a basic new approach, we have just

compared it with pure Pittsburgh GA in detail. More comparisons can be done in future works.

To compare the performance of SEA algorithm with Pittsburgh GA, we used six frequently used benchmarks: The first one is a 10% selection of KDDCUP99 dataset (MIT Lincoln Labs, 2007) and others are selected from University of California Irvine, Machine Learning Repository (Blake & Merz, 1998); these datasets are gathered from real experiments, so they can show efficiency of the algorithm in some real circumstances. Credit Approval (CRX), Glass Identification (Glass), Iris Plant (Iris), 1984 United States Congressional Voting Records Database (Vote), and Wine Recognition (Wine) datasets are selected as the most frequently used datasets so as to compare the results to some other related works. The extensive information about these datasets is mentioned in Table 1. Although KDDCUP99 data set has many classes of intrusion types, we consider their classes as Normal and Attack cases, similar to (Esposito et al, 2005), (Toosi & Kahani 2007), and (Mill & Inoue, 2004). General specifications of benchmarks are expressed in Table 1. The GA algorithm is implemented as described in (Hasanzadeh et al, 2004) with exactly the same parameters (expressed in Table 2).

Likewise (Hasanzadeh et al, 2004) & (Hasanzadeh & Bagheri, 2003), Fuzzy C-Mean clustering (Zimmermann, 1996) was used to define the fuzzy membership function for continuous attributes, and fuzzy singletons were defined for none-parametric attributes. The number of fuzzy sets for KDD99 features is 5 and for other problems, 3 fuzzy sets are created. The exact parameters of SEA algorithm are presented in Table 3.

The tests are done four-fold (Blake & Merz, 1998), i.e. the data was randomly divided into 4 sets and in each trial, one set was taken as test set, and the other 3 were used as training set. Each test is repeated for 20 times, and the average, minimum and maximum classification rates for training and tests results are depicted in subsection 5.2 tables. The stopping criterion of each run is an unchanging best fitness value during 5000 fitness function calls.

Also, the average number of fitness function calls to reach the highest classification accuracy and the average ratio between time and fitness function calls for each benchmark/algorithm is reported in subsection 5.3 as a measure of algorithms speed.

Dataset	Features count	Numeric Features	Nominal Features	Classes	Instances
KDD99	41	34	7	2	494021
CRX	15	6	9	2	690
Glass	10	9	1	6	214
Iris	4	4	0	3	150
Vote	16	0	16	2	435
Wine	13	13	0	3	178

Table 1. Datasets Specification

Parameter	Value
Maximum Population	200
Mutation Rate	0.7
Elitism Rate	0.2
Tournament Size	4

Table 2. Pittsburgh GA Parameters, as in (Hasanzdeh, 2003)

Parameter	Value
Population Size	1000
Selection Rate	6
Tournament Size	8
Random Creation Rate	4

Table 3. SEA Parameters

5.2 Accuracy comparison results

Tables 4 and 5 represent the classification rates of Pittsburgh Genetic Algorithm (PGA) and SEA training and test data, respectively. As presented there, SEA has found better rule-sets in compare with PGA in all cases on training sets and 4 of 5 on test sets.

Data Sets	PGA			SEA			Average SEA to PGA Improvement[1]
	Min	Max	Average	Min	Max	Average	
CRX	87.433	88.937	88.07	85.199	90.042	88.85	6.54 %
Glass	63.921	72.023	69.42	66.923	74.812	71.43	6.57 %
Iris	98.139	99.082	98.63	97.237	99.91	99.35	52.55 %
Vote	96.528	98.003	97.32	96.474	97.976	97.56	8.96 %
Wine	96.189	99.156	97.68	99.153	99.910	99.44	75.86 %
KDD99	87.433	88.937	88.07	85.199	90.042	88.85	6.54 %

Table 4. Average Classification Rate of PGA and SEA, Different Data Sets, on Training Data

Data Sets	PGA			SEA			Average SEA to PGA Improvement
	Min	Max	Average	Min	Max	Average	
CRX	83.746	87.654	85.27	84.888	86.476	85.58	2.1 %
Glass	63.377	71.370	68.62	67.878	74.008	70.68	6.56 %
Iris	91.85	99.923	94.95	91.805	99.909	95.57	12.28 %
Vote	91.789	98.661	95.31	92.611	97.972	95.04	-5.76 %
Wine	86.293	99.902	92.9	90.821	97.683	94.59	23.8 %
KDD99	84.263	87.654	94.36	85.156	85.16	99.31	87.77 %

Table 5. Average Classification Rate of PGA and SEA, Different Data Sets, on Test Data

Table 6 presents the best classification results of some other approaches ((Gomez et al, 2002), (Mendes et al, 2001), (Liu & Kwok, 2000), & (Rouwhorst & Engelbrecht, 2000)) which are reimplemented and tested by (Hasanzadeh, 2003) with similar settings as ours. As stated there, in cases that we had sufficient comparison data, SEA is better than other algorithms in all data sets.

Also Table 7 presents some other results from other papers that have used almost similar test specifications with that of ours. It must be emphasized that the test condition of these results does not fully comply that of ours, in some cases not exactly specified and in other slightly easier or harder. As depicted there, SEA is among the top 2 best results for all benchmarks.

[1] (SEA – PGA) / (100 – PGA)

Algorithm	CRX	Glass	Iris	Vote	Wine	KDD'99
Fuzzy Classifier with Expression Tree Representation (Gomez et al, 2002)			94.84	85.42	92.22	
Fuzzy Classifier with Co-Evolution (Mendes et al, 2001)	84.7		95.3			
Extended Genetic Rule Induction (Liu & Kwok, 2000)	77.39	72.43	95.3			
Evolution of Decision Trees (Rouwhorst & Engelbrecht, 2000)			94.1			
SEA	85.58	70.68	95.57	95.04	94.59	99.31

Table 6. Classification rate of some other algorithms with exactly similar settings in compare to SEA, from (Hasanzadeh, 2003).

Algorithm	CRX	Glass	Iris	Vote	Wine	KDD99
Fuzzy Kohonen Network (Lorenz et al, 1997)			91.33			
Fuzzy Classifier System (Lorenz et al, 1997)			96.00			
ID3 (Dong & Kothari, 2003)	81.16					
Naive Bayes (Dong & Kothari, 2003)	77.68					
Bayesian Network (Ezawa & Schuermann,1995)	86.5					
C 4.5 (Ezawa & Schuermann,1995)	85.5					
Discrimination Analysis (Ezawa & Schuermann,1995)	83.4					
Fuzzy Classifier System (Ishibuchi & Yamamoto, 2005)		68.22				
k-means (Guo et al, 2006)		63.08	92.67		68.54	
MLP Neural Network (Ueda, 2000)		70.3				
Hyper Sphere SVM (Liu et al, 2007)		62.15	95.68			
MLP Neural Network (Deodhare et al, 2007)					95.8	
Rule Extraction based on Grey Lattice Classification (Yamaguchi et al, 2005)					86.7	
Tree Support Vector Machine (Mill & Iune, 2004)						70.75
Array Support Vector Machine (Milll & Iune, 2004)						91.30
Fuzzy Rule Base with Linear Tree Genetic Representation (Dasgupta & Gonzalez, 2001)			94.5	94.7	93.9	
Average of above approaches	*82.84*	*65.93*	*94.03*	*94.7*	*86.23*	*81.02*
Best of above approaches	*86.5*	*70.3*	*96.00*	*94.7*	*95.8*	*91.30*
SEA	85.58	70.68	95.57	95.04	94.59	99.31

Table 7. Average Classification Rate of some other algorithms with almost similar test settings in compare to SEA.

5.2 Speed comparison results

Figures 5-10 depict the best fitness values over time for SEA and GA on the six stated datasets, averaged in all runs. As it is presented in the diagrams, SEA has found a better solution much faster than GA in all cases. Table 8 summarizes these results, and presents the average time taken to find the best result by each algorithm on each benchmark. As stated there, SEA has reached its best result notably faster than GA in all cases.

Also Figure 11 depicts the relation between number of fitness function calls and time for the two algorithms. Four curves show GA and SEA algorithms for CRX and Iris datasets which are, respectively, the largest and the smallest UCI ML Repository datasets used in this paper. The curves are almost linear with a slight trend toward taking more time for each fitness function call while the algorithms are proceeding. Thus, the progress of elite fitness can be considered through either time or fitness function calls in diagrams 5 to 10. Number of fitness function calls can be considered as a rough measure of the speed complexity of the algorithm as it removes the effects of programming details on algorithm speed.

Dataset	SEA	PGA	SEA to PGA Improvement
CRX	357	4650	92.32 %
Glass	164	280	41.42 %
Iris	40	633	93.68 %
Vote	89	1490	94.02 %
Wine	98	1710	94.26 %
KDD99	7012	54306	87.08 %

Table 8. Average time taken by SEA and Pittsburgh GA to find the best classifier on different benchmarks, in seconds.

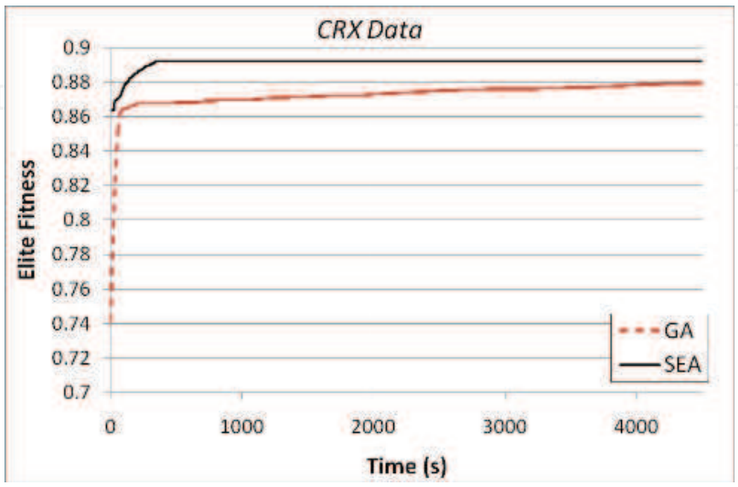


Fig. 5. CRX benchmark, average fitness of best rule set found by Pittsburgh GA and SEA over time.

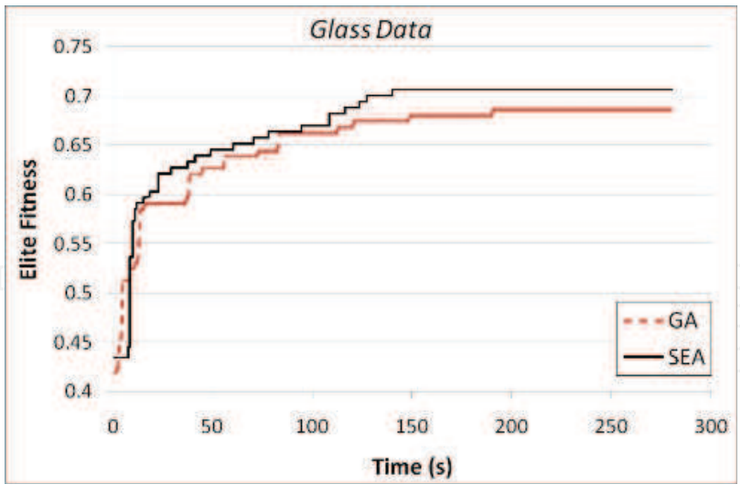


Fig. 6. Glass benchmark, average fitness of best rule set found by GA and SEA over time.

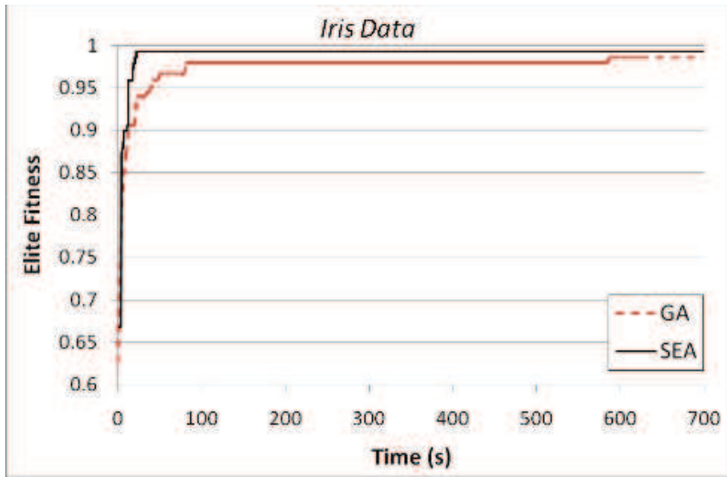


Fig. 7. Iris benchmark, fitness of best rule set found by Pittsburgh GA and SEA over time.

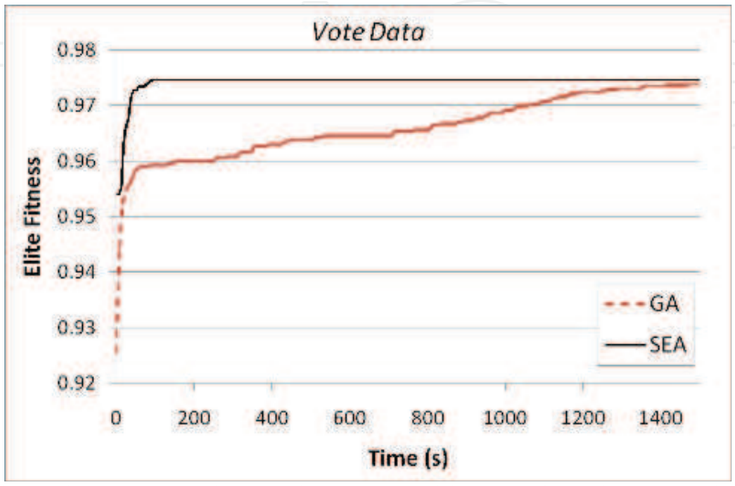


Fig. 8. Vote benchmark, fitness of best rule set found by Pittsburgh GA and SEA over time.

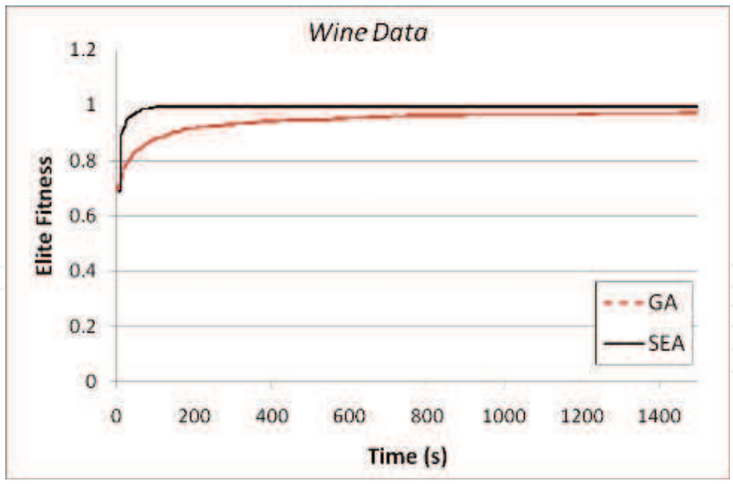


Fig. 9. Wine benchmark, fitness of best rule set found by Pittsburgh GA and SEA over time.

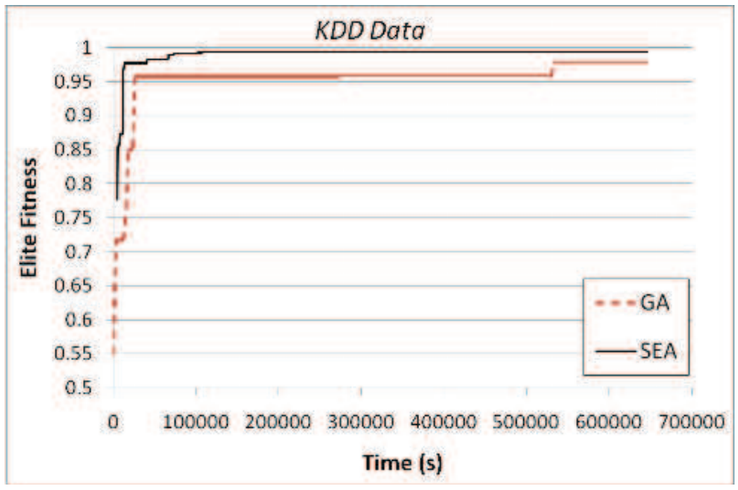


Fig. 10. KDD benchmark, fitness of best rule set found by Pittsburgh GA and SEA over time.

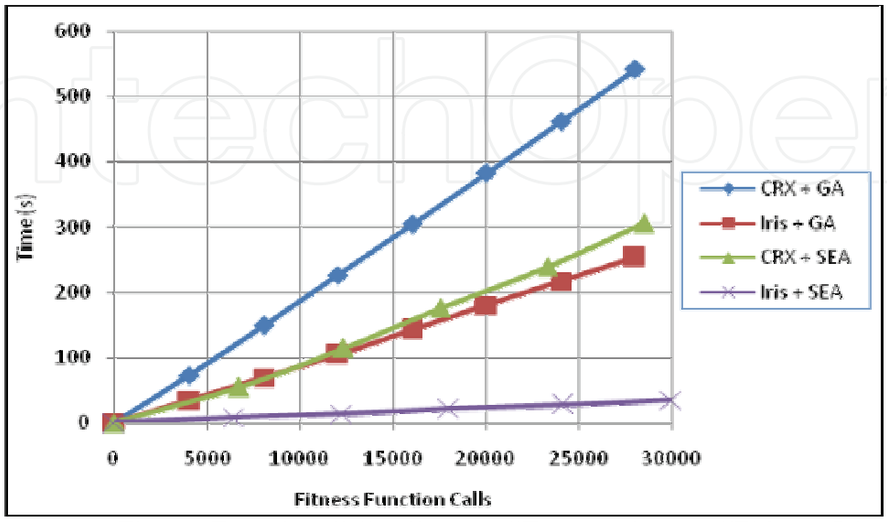


Fig 11. Time versus Fitness Function Calls, GA and SEA algorithms, CRX and Iris datasets.

6. Summary and concluding remarks

While the suitability of evolutionary approaches for generation of rule based classifier systems is shown in many different contributions, the structure and elements of this process are an important issue in design of a system that works efficiently. As stated in section 2, Michigan algorithm is faster and requires less memory in compare to Pittsburgh algorithm, but it has two very important problems that makes Pittsburgh the favorite one in many cases: First, the cooperation of single rules that are all evolved for better classification, regardless of other rule's behavior, will not necessarily result in a general good classifier; Second, some parts of the problem space might be neglected.

Pittsburgh evolutionary algorithm also has three problems that must be dealt with during an efficient implementation: First, how to recombine two rule-sets? While traditional sexual recombination operators splits the two parents and merges their parts, how should one know which rules of either rule-set (parents) must be extracted to be recombined to make a good combination. Second, what to do with the parasite rules? And the third question is how many rules must a rule-set have to get a small, but accurate classifier?

SEA algorithm uses symbiotic combination operator instead of common sexual recombination operator of GA, and provides a solution for the three above questions; it creates an offspring from two parents by combining all of their rules (genes), and adds the offspring to the gene pool only if it outperforms both its parents. Using this strategy, SEA avoids grouping separate rules before it makes sure that the group works better than the isolated ones, so it avoids garbage rules. It doesn't break any generated rule-set; therefore, it doesn't require a method to identify good working sub sets of two rule-sets. Also, as it grows the rule-sets only if growing results in better performance, the designer does not need to make a decision about chromosome sizes in advance.

Experimental results clearly comply with this hypothesis where SEA had 6 to 75 percent classification error reduction on training data in compare with Pittsburg GA and 2 to 87 percent on test data, except in one case which resulted in 6 percent more classification error. Moreover, this significant better accuracy was reached by 41 to 92 percent less computation time, in similar operating conditions.

As SEA is introduced as a basic algorithm to resolve the problems of Pittsburgh algorithm, we have just compared it in details with Pittsburgh GA, but some accuracy comparisons with algorithms from other families were also presented in section 6 and 7. Although some of these comparisons are not very fair as they were taken from different sources with slightly different test conditions, SEA presented very good comparison results to all of them as well.

Table 9 presents a features summary of SEA, Michigan, & Pittsburgh algorithms. As it is noted there, SEA stands between Michigan and Pittsburgh approaches from many viewpoints, collecting the positive points of both of them. SEA starts with light weight single rule individuals, as in Michigan, and gradually evolves them towards complete rule-set individuals, as in Pittsburgh. Due its growing size of individuals, it stands between Michigan and Pittsburgh in speed and memory complexity measures. Similar to Pittsburgh, it allows cooperation inside rule-sets but unlike Pittsburgh and similar to Michigan, this does not result in parasite rules, keeping rule-sets neat and accurate. Inheritance is done both on rule level and rule-set level as there is no distinction between rule and rule-sets. As the fitness of a rule-set is defined over all of its rules, a single rule that correctly covers a small uncovered portion of training samples can increase the credit of a rule-set and therefore is accepted and added to the rule-set, so, unlike Michigan approach there is no need to set specific credit to less frequently used training samples. And at last, in contrast to

Pittsburgh that blindly recombines two rule-sets, SEA combines two rule-sets only if this increases the overall recognition performance.
As next stages of this task, we can recommend an extra function that recognises rules that have redundant effects after symbiotic combinations. Also more specific representations and local optimization of rule-sets may result in better classification rates.

	Michigan	Pittsburgh	SEA
Individual	A single rule	A rule-set	Starts with single rules and reaches rule-sets
Selection and Evaluation	On each rule	On each rule-set	On each rule-sets
Rules Cooperation	None, Rules are rivals	Cooperative inside rule-sets, rival among rule-sets	Cooperative inside rule-sets, rival among rule-sets
Garbage Rules	Not Existing	Severely Existing	Not Existing
Computation Time	Least	Most	Between others
Memory Size	Least	Most	Between Others
Rule Optimization	Direct	Indirect	Both direct and indirect
Inheritance	Good Rules	Good Rule-Sets	Both good rules and rule-sets
Requires class credit assignments	Yes	No	No
Requires rule-set size specification	Yes	Yes / Controlled by a score function	No, controlled by accuracy.
Rule-Set recombination	None	Yes, but may result in lower accuracy	Yes, always results in higher accuracy.

Table 9. Feature Comparison of Michigan, Pittsburgh and SEA.

7. Acknowledgements

Authors wish to thank Mr. Pooya Esfandiar and Ms. Sima Lotfi for their help during implementation and testing of this task and Ms. Maryam Hasanzadeh for sharing the details of her implementation, test, and data sets.

8. References

Bagley, J.D. (1967). *The Behaviour of Adaptive Systems Which Employ Genetic and Correlation Algorithms,* PhD Dissertation, University of Michigan.

Barry, A., Holmes, J., & Llor, X. (2004). Data Mining using Learning Classifier Systems, In: *Applications of Learning Classifier Systems*, *Bull, L. (Ed.), 15-67, Springer,* ISBN:3540211098.

Blake C.L., Merz C.J. (1998). UCI Repository of machine learning databases, Irvine, CA: University of California, Department of Information and Computer Science. http://www.ics.uci.edu/~mlearn.

Chen, M.Y., Linkens, D.A., (2004). *Rule-base self-generation and simplification for data-driven fuzzy models*, Fuzzy Sets and Systems, Volume 142, Issue 2, 1 March 2004, Pages 243-265.

Chi-Ho Tsang; Sam Kwong; Hanli Wang, (2005). Anomaly intrusion detection using multi-objective genetic fuzzy system and agent-based evolutionary computation framework, in Proceedings of Fifth IEEE International Conference on Data Mining.

Corcoran, A.L., & Sen, S. (1994). *Using real-valued genetic algorithms to evolve rule sets for classification*. In Proceedings of the IEEE Conference on Evolutionary Computation, pages 120--124, 1994.

Cordon O., del Jesus M.J., Herrera F., (1998). *Genetic learning of fuzzy rule-based classification systems cooperating with fuzzy reasoning methods*, International Journal of Intelligent Systems 13 (10–11) 1025–1053.

Dasgupta, D. & Gonzalez, F., (2001). Evolving Complex Fuzzy Classifier Rules Using a Linear Tree Genetic Representation, In L. Spector, D. Whitley, D. Goldberg, E. Cantu-Paz, I. Parmee, and H. Beyer, editors, Proc. of the Int. Conf. on Genetic and Evolutionary Computation (GECCO-2001), pages 299-305. Morgan- Kaufmann, San Francisco, CA.

De Jong, K.A. , Spears, W., & Gordon, D.F. (1993). Using genetic algorithms for concept learning. *Machine Learning*, 13(2-3):155–188.

de la Iglesia B., Philpott M.S., Bagnall A.J., Rayward-Smith V.J., (2003), *Data Mining Rules Using Multi-Objective Evolutionary Algorithms*, in Proceedings of IEEE Congress on Evolutionary Computations, Vol. 3, pp 1552-1559.

Deb, K. (1991). *Binary and floating point function optimization using messy genetic algorithms* (IlliGAL Report No. 91004). Urbana: University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory.

Deodhare, D., Murty, M. N., and Vidyasagar, M., (2007). *A Unified Approach to Encoding and Classification using Bimodal Projection-based Features*, in Proceedings of the International Conference on Computing: Theory and Applications (ICCTA'07) pp. 348-354.

Dong, M. & Kothari, R., (2003). *Feature subset selection using a new definition of classifiability*, Pattern Recognition Letters 24 (2003) 1215–1225.

Eggermont J., Kok J.N., Koster W.A., (2003) *Genetic Programming for Data Classification: Refining the Search Space*, in Proceedings of the Fifteenth Belgium/Netherlands Conference on Artificial Intelligence, pp 123-130.

Esposito M., Mazzariello C., Oliviero F., Romano S. P., Sansone C., (2005). *Evaluating Pattern Recognition Techniques in Intrusion Detection Systems*, in Proceedings of the 7th International Workshop on Pattern Recognition in Information Systems (PRIS 2005) - 24-25 May 2005, Miami, FL (USA) pp. 144-153.

Ezawa, K. J., & Schuermann, T., (1995). *A Bayesian network Based Learning System: Architecture and Performance Comparison with Other Models*, in Proceedings of the European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty, pp 197 – 206.

Forrest S., Mitchell M. (1993). *Relative Building-block fitness and the Building-block Hypothesis*. In Whitley, D, ed. FOGA 2, Morgan Kaufmann, San Mateo, CA.

Freitas, A., *A survey of evolutionary algorithms for data mining and knowledge discovery*. In Advances in Evolutionary Computation. Springer- Verlag, 2001.

Goldberg, D.E.; Korb, B. & Deb, K. (1989) Messy Genetic Algorithms: Motivation, analysis, and first results. *Computer Systems*, 3, 5, 493-530.

Gomez, J., Dasgupta, D., (2002) *Evolving Fuzzy Classifiers for Intrusion Detection*, in Proceedings of the 2002 IEEE Workshop on Information Assurance.

Gomez, J., Gonzalez, F., Dasgupta, D., (2002). *Complete Expression Trees for Evolving Fuzzy Classifier Systems with Genetic Algorithms*, in Proceedings of the Evolutionary Computation Conference GECCO'02, 2002.

Gopalan J., Alhajj R., Barker J., (2006). *Discovering Accurate and Interesting Classification Rules Using Genetic Algorithm*, in Proceedings of the 2006 International Conference on Data Mining, pp. 389-395. June 26-29, 2006.

Gundo K.K., Alatas B., Karci A., (2004). *Mining Classification Rules by Using Genetic Algorithms with Non-random Initial Population and Uniform Operator*, Turkish Journal of Electrical Engineering and Computer Science, Vol.12, No. 1, 2004.

Guo, H.X., Zhu, K.J., Gao, S.W., & Liu, T., (2006). *An Improved Genetic k-means Algorithm for Optimal Clustering*, in Proceedings of Sixth IEEE International Conference on Data Mining - Workshops (ICDMW'06) pp. 793-797.

Harik, G.R. (1997). *Learning Gene Linkage to Efficiently Solve Problems of Bounded Difficulty Using Genetic Algorithm*, PhD Dissertation, University of Illinois at Urbana-Champaign, Urbana, Illinois.

Hasanzade M., (2003). *Fuzzy Intrusion Detection*, MS. Dissertation, Computer Engineering Department, Sharif University of Technology, Tehran, Iran, 2003.

Hasanzade, M., Bagheri, S., Lucas, C., (2004). *Discovering Fuzzy Classifiers by Genetic Algorithms*, in Proceedings of 4th international ICSC Symposium on Engineering of Intelligent Systems (EIS2004), 2004, Island of Madeira, Portugal.

Holland, John H. (1986). Escaping Brittleness: The Possibilities of General-Purpose Learning Algorithms Applied to Parallel Rule-Based Systems. In R. Michalski, J. Carbonell, and T. Mitchell (Eds.), Machine Learning: An Artificial Intelligence Approach (Vol. 2). Morgan Kaufmann Publishers, Los Altos, CA.

Ishibuchi H., Nakashima T., and Murata (1999). T., *A hybrid fuzzy genetics-based machine learning algorithm: Hybridization of Michigan approach and Pittsburgh approach,* in proceedings of IEEE, fuzzy IEEE.

Ishibuchi H., Yamamoto T., (2002). *Fuzzy rule selection by data mining criteria and genetic algorithms,* in Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2002), pp. 399-406, New York, July 9-13.

Ishibuchi H., Yamamoto T., (2004). *Fuzzy Rule Selection by Multi-Objective Genetic Local Search Algorithms and Rule Evaluation Measures in Data Mining,* Fuzzy Sets and Systems, Vol. 141, no. 1, pp. 59-88, January 2004.

Ishibuchi, H. &, Yamamoto, T., (2004). *Rule Weight Specification in Fuzzy Rule-Based Classification Systems* , IEEE Transactions on Fuzzy Systems, vol. 13, no. 4, August 2005.

Janikow, C.Z. (1993) A knowledge-intensive genetic algorithm for supervised learning. *Machine Learning*, 13(2-3):180–228.

Liu J.J., & Kwok J.T. (2000). *An Extended Genetic Rule Induction Algorithm*, in Proceedings of IEEE Congress on Evolutionary Computation (CEC-2000). La Jolla, CA, USA. July 2000.

Liu, S., Liu, Y., Wang, B., and Feng, X., (2007). *An Improved Hyper-sphere Support Vector Machine*, in Proceedings of the Third International Conference on Natural Computation (ICNC 2007) pp. 497-500.

Lopes C., Pacheco M, Vellasco M, Passos E, (1999). *Rule-Evolver: An Evolutionary Approach For Data Mining*, in Proceedings of the 7th International Workshop on Rough Sets, Fuzzy Sets, Data Mining, and Granular-Soft Computing, RSFDGrC'99, pp 458-462.

Lorenz, A., Blum, M., Ermert, H., & Senge, T., (1997). *Comparison of Different Neuro-Fuzzy Classification Systems for the Detection of Prostate Cancer in Ultrasonic Images*, in Proceedings of the IEEE Ultrasonics Symposium, 1997, Volume: 2, pp 1201-1204.

Maynard Smith, J., Szathmary, E. *The Major Transitions in Evolution*, WH Freeman: Oxford UK, 1995.

Merezhkovsky, K. S. (1909), *The Theory of Two Plasms as the Basis of Symbiogenesis, a New Study or the Origins of Organisms*. In Proceedings of the Studies of the Imperial Kazan University, Publishing Office of the Imperial University, (In Russian).

Mendes, R., R., F., Voznika, F., de B., Freitas, A., A., Nievola, J. C., (2001). *Discovering Fuzzy Classification Rules with Genetic Programming and Co-Evolution*, In Principles of Data

Mining and Knowledge Discovery (Proc. 5th European Conference PKDD 2001) – Lecture Notes in Artificial Intelligence, Springer-Verlag.

Mill J., Inoue A. (2004). *Support Vector Classifiers and Network Intrusion Detection*, in Proceedings of IEEE Conference on Fuzzy Systems 2004, Vol 1. pp 407-410.

MIT Lincoln Labs, (2007). KDD CUP 99 DARPA Intrusion Detection Dataset, http://kdd.ics.uci.edu/databases/kddcup99.

Mitchell, M. (1999). *An Introduction to Genetic Algorithms,* MIT Press, 0−262−13316−4, London, England.

Potter, M.A., De Jong, K.A. (1994). A Cooperative Coevolutionary Approach to Function Optimization. *In: Parallel Problem Solving from Nature (PPSN III)*, Y. Davidor, H.-P. Schwefel and R. Manner (Eds.). Berlin: Springer-Verlag, 249-257.

Riquelme J.S., Toro J.C., Aguilar-Ruiz M., (2003), *Evolutionary Learning of Hierarchical Decision Rules*, in IEEE Transactions on Systems, Man, and Cybernetics, Vol. 33, Issue 2, pp 324-334.

Rouwhorst, S.E., Engelbrecht A.P., (2000). *Searching the Forest: Using Decision Tree as Building Blocks for Evolutionary Search in Classification*. In Proceedings of IEEE Congress on Evolutionary Computation (CEC-2000), 633-638. La Jolla, CA, USA. July 2000.

Sen, S., Knight, L., & Legg, K. (1997). Prototype based supervised concept learning using genetic algorithms. In D. Dasgupta and Z. Michalewicz, editors, *Evolutionary Algorithms in Engineering Applications*, pages 223–239. Springer.

Smith, S. F. (1980). *A Learning System Based on Genetic Adaptive Algorithms*, PhD Thesis, University of Pittsburgh.

Smith, S. F. (1983). Flexible Learning of Problem Solving Heuristics Through Adaptive Search, Proc. 8th IJCAI, August 1983.

Tan K.C., Yu Q., Heng C.M., Lee T.H., (2003). *Evolutionary computing for knowledge discovery in medical diagnosis*, Artificial Intelligence in Medicine 27, pp.129-154.

Teng M., Xiong F., Wang R., Wu Z., *Using genetic algorithm for weighted fuzzy rule-based system*, in Proceedings of Fifth World Congress on Intelligent Control and Automation, 2004.

Toosi A.N., Kahani M. (2007). *A New Approach to Intrusion Detection Based on an Evolutionary Soft Computing Model Using Neuro-Fuzzy Classifiers,* Computer Communications, Vol 30, 2201-2212.

Ueda, N., (2000). *Optimal Linear Combination of Neural Networks for Improving Classification Performance,* IEEE Transactions on Pattern Analysis and Machine Learning, vol. 22, no. 2, February 2000.

Watson, R.A. & Pollack, J.B. (1999), Incremental Commitment in Genetic Algorithms, *Proceedings of GECCO'99.*, Morgan Kaufmann, 710-717.

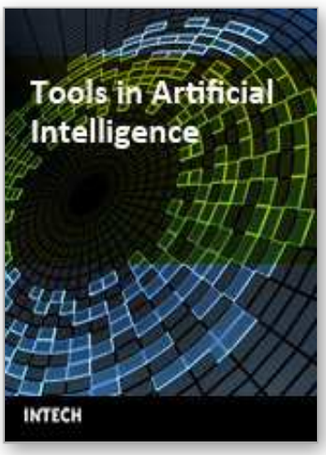
Watson, R.A., Pollack, J.B. (2000). *Symbiotic Combination as an Alternative to Sexual Recombination in Genetic Algorithms*, in Proceedings of Parallel Problem Solving from Nature (PPSN VI).

Wilson, S. (1987). Classifier systems and the Animat problem. *Machine Learning*, 2:199–228.

Yamaguchi, D., Li, G.D., Mizutani, K., and Akabane, T., (2005). *Decision Rule Extraction and Reduction Based on Grey Lattice Classification*, in Proceedings of the Fourth International Conference on Machine Learning and Applications, 2005, 15-17 Dec. 2005.

Zimmermann, H., J., (1995). *Fuzzy Set Theory and Its Application*, Kluwer Academic Publishers.

Zhu F., Guan S.U., (2004). *Ordered Incremental Training with Genetic Algorithms*, International Journal of Intelligent Systems, Volume 19, Issue 12 , pp 1239-1256.



Tools in Artificial Intelligence

Edited by Paula Fritzsche

ISBN 978-953-7619-03-9
Hard cover, 488 pages
Publisher InTech
Published online 01, August, 2008
Published in print edition August, 2008

This book offers in 27 chapters a collection of all the technical aspects of specifying, developing, and evaluating the theoretical underpinnings and applied mechanisms of AI tools. Topics covered include neural networks, fuzzy controls, decision trees, rule-based systems, data mining, genetic algorithm and agent systems, among many others. The goal of this book is to show some potential applications and give a partial picture of the current state-of-the-art of AI. Also, it is useful to inspire some future research ideas by identifying potential research directions. It is dedicated to students, researchers and practitioners in this area or in related fields.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen