Information Technology and Management 6, 315–331, 2005

Efficient Genetic Algorithm Based Data Mining Using Feature Selection with Hausdorff Distance

RIYAZ SIKORA rsikora@uta.edu
Department of Information Systems & OM, University of Texas at Arlington, P.O. Box 19437, Arlington, TX 76019

SELWYN PIRAMUTHU
Department of Decision and Information Sciences, University of Florida, Gainesville, FL 32611-7169

Abstract. The development of powerful computers and faster input/output devices coupled with the need for storing and analyzing data have resulted in massive databases (of the order of terabytes). Such volumes of data clearly overwhelm more traditional data analysis methods. A new generation of tools and techniques are needed for finding interesting patterns in the data and discovering useful knowledge. In this paper we present the design of more effective and efficient genetic algorithm based data mining techniques that use the concepts of self-adaptive feature selection together with a wrapper feature selection method based on Hausdorff distance measure.

Keywords: rule learning, knowledge discovery, data mining, evolutionary algorithms, feature selection, genetic algorithms

1. Introduction

The development of powerful computers and faster input/output devices coupled with the need for storing and analyzing data have resulted in massive databases (of the order of terabytes). Our ability to produce and store such voluminous data has far outpaced our ability to analyze and interpret the data, and derive useful knowledge from it. Large databases these days have millions of records and each record may have hundreds or even thousands of fields. For example, in the business world, one of the largest databases was created by Wal-Mart, which handles over 20 million transactions a day [6]. Similar instances can be found in databases created by health care companies, oil exploration firms, database marketing, and scientific research consortiums, just to name a few.

Such volumes of data clearly overwhelm more traditional data analysis methods. A new generation of tools and techniques are needed to find interesting patterns in the data and discover useful knowledge. These tools and techniques and their applications are the subject of the field of Knowledge Discovery and Data Mining [26]. Successful development of effective and efficient data mining algorithms can also provide enormous benefits to an organization from the business standpoint. Benefits include reduced costs due to more accurate control, more accurate future predictions, more effective fault

detection and prediction, fraud detection and control, and automation of repetitive human tasks.

In this paper we present the design of more effective and efficient genetic algorithm (GA) [29,31] based data mining techniques that use the concepts of self-adaptive feature selection together with a wrapper feature selection method based on Hausdorff distance measure. The data mining algorithms developed are tested on a real world problem. The problem is related to the control of a chemical process at an Eastman Kodak facility in Longview, TX, producing a certain chemical product, with about 30 process variables. In the process of producing the product, an undesirable byproduct is produced which is not measured directly. To remove this byproduct, an expensive chemical is added in just sufficient quantities. The problem is to change the controllable process variables (nine out of the thirty variables) so that the usage of the expensive chemical is minimized. Since there are no theoretical formulae linking the process variables with the amount of the product produced, the only way to solve this problem is to induce the relationships based on a set of actual plant readings. All the plant readings were normalized to be between 0.0 and 1.0 and the goal for the plant manager was to maintain the usage of the expensive chemical to under 0.35 at all times.

The problem was formulated as a single concept learning problem by considering the examples where the quantity of the expensive chemical used was less than 0.35 as being positive examples and the rest as negative examples. The goal of the data mining algorithms would be to discover or "learn" concepts that would cover or explain all the positive examples without covering any negative examples. In other words, the data mining algorithms or "learning" algorithms would discover relationships among the control variables that would reduce the usage of the expensive chemical. The available data set is divided into two subsets for training and testing purposes. The data mining algorithms use the first sub-set, also known as *training examples*, to learn or discover the required concepts. The learned concepts are then evaluated using the holdout sub-set, also known as *testing examples*.

There are many data mining algorithms currently in use. Most of them can be classified in one of the following categories: Decision trees and rules [14,49] nonlinear regression and classification methods [28,23], example-based methods [20,35], probabilistic graphical dependency models [47,58], and relational learning models [21]. Over the years genetic algorithms have been successfully applied in learning tasks in different domains, like chemical process control [52], financial classification [53], manufacturing scheduling [40], robot control [54], etc. There has also been some work done related to developing hybrid learning systems involving genetic algorithms [9,57]. Evolutionary algorithms (including GAs) have also been recently used for data mining tasks [5,12,32,44].

2. Feature selection

Feature selection is the problem of choosing a small subset of features that ideally is necessary and sufficient to describe the target concept [37]. A goal of feature selection is

to avoid selecting too many or too few features than is necessary. If too few features are selected, there is a good chance that the information content in this set of features is low. On the other hand, if too many (irrelevant) features are selected, the effects due to noise present in (most real-world) data may overshadow the information present. Hence, this is a tradeoff that must be addressed by any feature selection method.

The marginal benefit resulting from the presence of a feature in a given set plays an important role. A given feature might provide more information when present with certain other feature(s) than when considered by itself. [18,22], and [55], among others, have shown the importance of selecting features as a set, rather than selecting the best features to form the (supposedly) best set. They have shown that the best individual features do not necessarily constitute the best set of features.

However, in most real-world situations, the best set of features or the number (n) of features in such a set is not known. Currently, there is no means to obtain the value of n, which depends partially on the objective of interest. Even assuming that n is known, it is extremely difficult to obtain the best set of n features since not all n of these features may be present in the data comprising the available set of features.

There have been many approaches to feature selection based on a variety of techniques, such as statistical [38], geometrical [24], information-theoretic measures [1,10, 15], mathematical programming [13], neurofuzzy [11], Receiver Operating Curves [17], discretization [42], among others. Several researchers have also used evolutionary algorithms for feature selection by using a classifier as a fitness function [8,25,33,36,48,56, 57]. GAs have also been used in feature selection for creation of ensemble classifiers [30,46].

Feature selection has been traditionally used in data mining applications as part of the data cleaning and/or pre-processing step where the actual extraction and learning of knowledge or patterns is done after a suitable set of features is extracted. If the feature selection is independent of the learning algorithm it is said to use a *filter* approach. [39] presents one such filter approach to feature selection using genetic algorithms If the feature selection method works in conjunction with the learning algorithm it is using a *wrapper* approach [34,59]. The problem with the filter approach is that the optimal set of features may not be independent of the learning algorithm or classifier. The wrapper approach provides a better approach, however it is computationally expensive as each candidate feature subset has to be evaluated by executing a learning algorithm on that subset.

When used with GAs, the wrapper approaches become even more prohibitively expensive [39]. [50] presents an approach of simultaneously doing feature selection and optimizing feature weights using a GA. In this article we present a wrapper approach to feature selection using a GA as the learning algorithm and show how the computational complexity of the approach can be reduced by incorporating a second self-adaptive feature selection approach where the learning and feature selection are done simultaneously.

The feature selection concept presented in this article is related to two aspects of work done earlier: self-adaptation and use of non-coding material in the chromosome structure (called *introns*) motivated by the existence of non-encoding DNA in biological

systems. In biological systems an intron is a portion of the DNA that is not transcribed into proteins. Introns can become an important part of the evolution process by providing a buffer against the destructive effects of the genetic algorithm. At the same time introns have been shown to be useful as a source of symbols that can be effectively used to evolve new behaviors through subsequent evolution [41,45].

Self adaptation [3] refers to the technique of allowing characteristics of the search to evolve during the search rather than be specified by the user. Most of the work done on self-adaptation has focused on choices related to search operators. Aspects of these choices are encoded along with each member of the population and they are allowed to vary and adapt on an individual basis. One of the most common traits to be self-adapted has been the mutation rate [7,27,51]. Others have included crossover [4] and inversion operators [16].

3. Data mining with GA

In this section we present the design of a genetic algorithm for rule learning in a data mining application. We discuss the five important components needed to design a GA: (1) the representation format, (2) the fitness function, (3) the selection and reproduction operators, (4) various parameter values, (5) a method for generating initial population and a stopping criteria.

The representation of a concept or a classifier used by the GA is that of a disjunctive normal form. A concept is represented as $C = C_1 \vee C_2 \vee \ldots \vee C_p$, where each disjunct C_i (henceforth called a rule) is a conjunction of conditions,

$$= \big(\xi_{1,1} \,\&\, \ldots \,\&\, \xi_{k_1,1}\big) V (\xi_{1,2} \,\&\, \ldots \,\&\, \xi_{k_2,2}) V \ldots V \big(\xi_{1,p} \,\&\, \ldots \,\&\, \xi_{k_p,p}\big).$$

The above concept C is said to have a size of p (which we refer to as the rule size). An ideal concept would cover all the positive examples without covering any of the negative examples. In order to handle *continuous variables* each condition $\xi_{j,i}$ is in the form of a closed interval $[a_i b_i]$, i.e.,

$$\begin{aligned} \xi_{j,i} = (a_i \le X_j \le b_i), \;\; &\text{if } a_i \le b_i \\ \text{or } (a_i \ge X_j \ge b_i), \quad &\text{if } a_i \ge b_i \end{aligned}$$

The following example will help explain the above representation. Consider the following concept C for a problem involving two attributes X_1 and X_2:

$$\begin{aligned} C &= [(-10)(0.40.9)] \vee [(00.5)(-0.20.9)] \\ &= [(-1 \le X_1 \le 0) \text{ and } (0.4 \le X_2 \le 0.9)] \text{ OR} \\ &\quad [(0 \le X_1 \le 0.5) \text{ and } (-0.2 \le X_2 \le 0.9)] \end{aligned}$$

In terms of the above representation, $p = 2, k_1 = k_2 = 2, C = C_1 \vee C_2, C_1 = (\xi_{1,1} \,\&\, \xi_{2,1})$, $C_2 = (\xi_{1,2} \,\&\, \xi_{2,2})$, $\xi_{1,1} = (-1 \le X_1 \le 0)$, $\xi_{2,1} = (0.4 \le X_2 \le 0.9)$, $\xi_{1,2} = (0 \le X_1 \le 0.5)$, and $\xi_{2,2} = (-0.2 \le X_2 \le 0.9)$.

Each member of the population in the GA is a single disjunct $C_i = (\xi_{1,i} \,\&\, \ldots \,\&\, \xi_{ki,i})$. Using the above example, C_1 and C_2 would be individual members (chromosomes) in the population represented as a string of four numbers as shown below:

$$\begin{array}{lcccc} C_1: & -1 & 0 & 0.4 & 0.9 \\ C_2: & 0 & 0.5 & -0.2 & 0.9 \end{array}$$

Note that the value of K_i for each i in the above representation is same as the total number of attributes. The selection of relevant attributes is done implicitly by the GA, by learning condition $\xi_{j,i}$ that represent the entire domain of the irrelevant attribute X_j. Later on we will present a feature selection method where the value of K_i will be simultaneously learned by the GA along with the rules.

The genetic algorithm tries to find the best possible disjunct that explains as many positive examples of the concept as possible while covering as few of the negative examples as possible. At each generation it retains the best disjunct and replaces the rest through the application of the genetic operators. After the genetic algorithm converges, the best disjunct found is retained and the positive examples it covers are removed. The process is repeated until all the remaining positive examples in the data set are covered. The final rule or concept is then the disjunct of all the disjuncts found. This procedure for searching the instance space (I-space) is called *explanation based filtering.*

Most of the parameter values for the GA were selected based on the common values often found in the literature. Note that the main purpose of this paper is to highlight a methodology for improving the performance of the basic GA and make it more robust for data mining applications. Since the same basic GA is used in all the four methods discussed, improving the performance of the basic GA by tweaking its various parameter values will not have any effect on the conclusions reached in this paper about the relative performance of the methods.

The GA implemented here uses a uniform crossover operator with a probability of crossover equal to 0.7. The interval range for each attribute on the rule is considered as a single entity for the purpose of crossover. The mutation operator used can be thought of as a specialization/generalization operator, which either increases or decreases (with equal probability) the interval range of an attribute by either increasing or decreasing the upper or lower interval limit with equal probability of 0.1. The reproduction operator uses a tournament selection with a size of 2. All the population members, except for the best one, are replaced from one generation to the next by applying the genetic operators. Please refer to [29] for detailed information about the implementation of the above operators.

The fitness function looks at the number of positive and negative examples covered by the rule but it also assigns partial credit for the number of attribute intervals on that rule that match the corresponding attribute values on a positive training example. Specifically, the fitness function is given by

$$F = \alpha + C\nu(p - n), \tag{1}$$

where α is the total number of partial matches, C is a constant, ν is the number of attributes, p is the number of positive examples covered by the rule, and n is the number of negative examples covered by the rule. Note that the partial credit portion of the fitness functions plays an important role in the initial generations in guiding the concepts that are being developed towards covering positive examples. As the concepts start covering more and more positive examples, the second term in the fitness function becomes correspondingly more dominant. The fitness function thus behaves like an adaptive function.

The initial population is created by generating disjuncts using random numbers for the lower and upper interval limits corresponding to each attribute. A population size of 100 is used and the terminating criterion for the GA is the non-improvement in the fitness value of the best individual in 10 generations. Note that since the fitness value can only improve by covering more positive examples and since there are finite number of positive examples to cover, it follows that the GA terminates after finite number of generations. All the algorithms used in this paper were implemented in C++ on a Sun workstation running SunOS 5.7.

4. Feature selection as self-adaptation

The GA presented above is modified as follows to incorporate the self-adaptive feature selection method. Each population member contains, in addition to the disjunct, a binary vector for feature selection that also evolves alongside the disjunct. A feature is selected if the corresponding bit in the selection vector is 1. For example, if we have five attributes in the original feature set a typical rule (disjunct) represented in the new approach would look like the following:

$$\begin{array}{ccccc} X_1 & X_2 & X_3 & X_4 & X_5 \\ ((0\ 0.1) & (0.12\ 0.24) & (0.23\ 0.5) & (0.4\ 0.7) & (0.2\ 0.87)) \\ 0 & 1 & 0 & 1 & 1 \end{array}$$

Since only X_2, X_4, and X_5 have a corresponding 1 on the selection vector, the rule becomes:

$$\text{IF}(0.12 \leq X_2 \leq 0.24) \text{ and } (0.4 \leq X_4 \leq 0.7) \text{ and } (0.2 \leq X_5 \leq 0.87)$$

The uniform crossover operator is applied to the trio of values (the interval limits and the selection bit) for each attribute instead of the pair of values as in the last section. The mutation operator flips the binary digits on the selection vector in addition to changing the pair of numbers for each attribute as explained in the last section. The initial population is created as before with the interval pairs for each attribute on a member created with uniform distribution and the binary selection vector randomly generated with a probability of *Fselect* for selecting a particular feature (i.e., a 1 appearing on the selection vector corresponding to that feature). The fitness function remains the same as before except that the selection vector is also used in deciding which attributes are

considered for fitness evaluation. Since the fitness evaluation now also depends on the feature subset selected we can hypothesize that this would start evolutionary pressures for good features to be selected.

Note that an attribute's interval limits on a rule can change due to crossover or mutation even when that feature is not selected in the rule. This is similar to the concept of introns mentioned earlier where non-coding genes are allowed to propagate and evolve in the hope that in a later generation they might be found to be useful. The interval limits of the attributes not selected in a rule can be thought of as introns. Since these are not used in the fitness evaluation they do not affect the computation time. The only additional resource they consume is the memory storage. However, since the GA used is a fixed length GA the use of introns in this case does not lead to the problems of bloat so often associated with the use of introns in the genetic programming community.

In the next section we present a wrapper method for feature selection that can be used with the GA designed above.

5. Hausdorff distance measure and feature selection

5.1. Hausdorff distance

The Hausdorff distance measure is widely used in computer vision and graphics applications due to its excellent discriminant properties. It has, however, not received much attention in the feature selection literature. The Hausdorff distance [43] is a measure of similarity, with respect to their position in metric space, of two non-empty compact sets A and B. It measures the relative position of each point in a set with that of another set. Let $X_1 = \{x_{11}, x_{12}, \ldots, x_{1m}\}$ and $X_2 = \{x_{21}, x_{22}, \ldots, x_{2n}\}$ be two finite point sets and d a distance over this space. The measure d can be any distance including the 1-norm,[1] the Euclidean norm, as well as simple difference between corresponding coordinates in each dimension, among others. The Hausdorff distance is defined as follows:

$$\forall x_1 \in X_1,\ D(x_1, X_2) = \min_{x2 \in X2} \{d(x_1, x_2)\} \tag{2}$$

$$h(X_1, X_2) = \max_{x1 \in X1} \{D(x_1, X_2)\} \tag{3}$$

$$H(X_1, X_2) = \max \{h(X_1, X_2), h(X_2, X_1)\} \tag{4}$$

Where, $h(X_1, X_2)$ is the directed Hausdorff distance from X_1 to X_2. It identifies the point $x^* \in X_1$ that is farthest (using a pre-specified norm) from any point in X_2 and measures the distance from x^* to its nearest neighbor in X_2. Essentially, $h(X_1, X_2)$ ranks each point in X_1 based on its distance from the nearest point in X_2 and then uses the largest ranked such point (x^*, the point in X_1 farthest away from X_2) as the distance. If $h(X_1, X_2) = A$, then each point in X_1 has at least one point in X_2 within radius of A. For smaller values of A, X_1 is nearly included in X_2. The Hausdorff distance itself, $H(X_1, X_2)$ is the maximum of the directed Hausdorff distances $h(X_1, X_2)$ and

$h(X_2, X_1).H(X_1, X_2)$ can be calculated in $O(m, n)$ for two point sets of size m and n respectively. [2] improve this to $O((m + n) \log(m + n))$.

The Hausdorff distance H is a metric over the set of all closed, bounded sets [19]. Being a true distance, it also obeys the properties of identity, symmetry, and triangle inequality. In the context of classification, it follows that the description of a concept is identical only to its own description, the order of comparing different concepts does not matter, and the descriptions of two different concepts cannot be similar to some third concept.

5.2. Feature selection with Hausdorff (HS) distance applied to GA

The following algorithm is used for applying feature selection using Hausdorff distance measure to the GA.

Algorithm HS + GA (feature selection with Hausdorff distance applied to GA):
Variables in data: $v_1, v_2, \ldots, v_k$
S = set of all input variables = $\emptyset$

1. Set $i = 1$; While $i < k + 1$, do
   a. Calculate $H_i(X_1, X_2)$ for v_i.
   b. $i = i + 1$.
   c. Go to 1(a).
2. Sort $H_i(.,.)$, in descending order, with the corresponding variables $(v_1, \ldots, v_k)$.
3. Set $j = 0$; Until stopping criterion is met, do
   a. $\delta = \Delta_j$; $S = S + (v_{j+1}, v_{j+2} \ldots, v_{j+\delta})$
   b. Let GA induce the best concept with input S.
   c. Evaluate quality of the learned concept.
   d. $j = j + \delta$.
   e. Go to 3(a).
4. Return the final concept learned.

Step 1 calculates the Hausdorff distance $H_i(X_1, X_2)$ between positive and negative examples, individually for each of the k variables (features) in the data set. This is followed by sorting the H_i values, and the corresponding variables are noted $(v_1, \ldots, v_k)$ in Step 2. This is followed by evaluation of the feature set by learning the best concept. We use the GA, discussed above, for generating the learned concept. The concepts are generated iteratively as more variables are added to the data set, in ascending order, based on their corresponding H_i values. The quality of the concept thus generated is evaluated using the fitness function in (1). The algorithm stops when a pre-specified stopping criterion

(no improvement in classification accuracy) is reached, and the final concept learned is returned.

In the next section we discuss the problem domain and present the experimental results.

6. Experimental results

6.1. Experimental design

The data set had 572 instances, of which 355 were positive examples and 217 were negative examples. It was randomly broken up into 10 pairs of training and testing (or holdout) sets with 344 and 228 examples respectively (each with the same proportion of positive and negative examples). Three different problem variations were created from the above data set. In the first case only the 9 controllable variables were used. In the second case all the 30 variables were used, and for the last case 20 additional variables were added by recombining some of the original 30 variables to create a data set with 50 variables. Having these three different sizes of essentially the same problem allows us to test the performance of the system as the size of the problem domain increases and study its scale-up properties, something very crucial for data mining applications. The last problem variation also allows us to test the effectiveness with which the data mining system can detect the irrelevant attributes. Typical data mining applications involve a lot of irrelevant attributes and it is widely recognized that around 80% of the resources in such applications are spent on cleaning and preprocessing the data.

When applying the Hausdorff distance measure for feature selection the variables are added as follows: For the 9 variable problem the variables are added in increments of 5 (i.e., $\Delta_0 = 5$ and $\Delta_5 = 4$). For the 30 and 50 variable problems, we start with 5 and 10 variables and then add in increments of 10 (i.e., $\Delta_0 = 5$, $\Delta_5 = 5$, $\Delta_{10} = 10$, etc.).

6.2. Results and discussion

The detailed results for the 9 variable, 30 variable, and the 50 variable problems are presented below in Tables 1–3. Each experiment was repeated 5 times with a different random number seed. The values in each cell of the tables are the average of those 5 trials. To evaluate the effect of feature selection on the performance of the system, 4 different versions of the system were tested. In the first case, a simple GA (as described in Section 3) is used to learn the concepts. In the second version (GAFS), the self-adaptive technique of explicit feature selection (as discussed in Section 4) is applied to the GA. In the third version (HS + GA), feature selection with Hausdorff distance (as discussed in Section 5) is used with the GA. And, finally, both the techniques of feature selection are combined in (HS + GAFS). For benchmarking purposes we also present the results of using a decision tree learning method C4.5 [49] on the data sets. Three criteria are

Table 1
Results for 9 variable problem.

	C4.5			Simple GA w/o feature selection (GA)			GA with simultaneous FS (GAFS)			Hausdorff Dist. FS applied to GA (HS + GA)			Combining both FS techniques (HS + GAFS)		
Data set	(%)	Sec.	Size	(%)	Sec.	Size	(%)	Sec.	Size	(%)	Sec.	Size	(%)	Sec.	Size
1	78.6	14.72	43	78.77	20.67	4	78.86	9.61	3.2	78.77	32.51	4	78.86	21.45	3.2
2	84.3	16.42	49	83.63	24.52	4	82.10	9.97	3.6	83.63	39.77	4	82.10	25.22	3.6
3	81.7	12.03	47	81.32	22.24	4.2	80.52	10.36	3.8	81.32	33.83	4.2	80.52	21.95	3.8
4	85.2	12.8	47	80.26	21.78	3.6	80.52	8.89	3.6	80.26	35.05	3.6	80.52	22.16	3.6
5	85.2	18.87	67	83.95	21.83	3.4	83.49	10.84	3.8	83.95	38.26	3.4	83.49	27.27	3.8
6	84.3	16.24	53	85.52	24.71	3.8	84.02	9.78	3.8	85.52	38.11	3.8	84.02	23.18	3.8
7	82.5	13.45	41	79.56	21.14	4	80.61	11.90	4	79.56	35.23	4	80.61	25.99	4
8	84.3	14.52	55	81.32	21.04	3.8	80.79	10.94	3.4	81.32	35.40	3.8	80.79	25.30	3.4
9	83	12.78	45	82.55	23.11	3.6	81.14	8.54	3.4	82.55	35.70	3.6	81.14	21.13	3.4
10	80.8	17.23	49	81.4	21.31	3.6	81.75	9.97	3.4	81.4	36.05	3.6	81.75	24.72	3.4
Avg.	**82.99**	**14.91**	**49.6**	**81.73**	**22.24**	**3.8**	**81.38**	**10.08**	**3.6**	**81.73**	**35.99**	**3.8**	**81.38**	**23.84**	**3.6**
Std.	*2.13*	*2.23*	*7.43*	*2.02*	*1.43*	*0.25*	*1.52*	*0.99*	*0.25*	*2.02*	*2.18*	*0.25*	*1.52*	*2.14*	*0.25*
							Paired t-test (2 tail)								
$P_{C4.5}$				n.s.	5.8E–6	1E–8	0.01	7.7E–5	1E–8	n.s.	5E–11	1E–8	0.01	7.6E–8	1E–8
P_{GA}							n.s.	1.3E–8	0.09	n.s.	4E–10	n.s.	n.s.	0.09	0.09
P_{GAFS}										n.s.	6E–11	0.09	n.s.	3.9E–10	n.s.
P_{HS+GA}													n.s.	3.9E–10	0.09

used for doing the comparisons. The prediction accuracy of the concept learned on the hold-out testing sample, the computation time, and the simplicity of the concept learned as measured by the number of rules in the concept. For fairness of comparison, the population size of the GA was fixed at 100 for all the three problem sizes so that the number of function evaluations across the problem sizes would remain more or less the same. The average and standard deviation values are provided together with the results of Paired t-test for statistical test of significance. Table 4 summarizes the average results. Figures 1–3 depict the comparative performance of the four methods on the three criteria.

Some of the major findings can be summarized as follows:

- Performance of the simple GA quickly deteriorates as the problem size increases. Its prediction accuracy plummets at the same time its computational time and the complexity of the concept increase substantially. For C4.5, the performance is more stable for the prediction accuracy and the complexity of the concept. However, its computational time also increases substantially with the increase in problem size. Note that the GA is much better at producing simpler and compact concepts as compared to the C4.5. This might be especially important for managerial purposes as the concepts produced by the GA are easy to understand and hence easier to implement.
- When self-adaptive feature selection is added to the simple GA (GAFS), the performance in terms of computation time and the simplicity of the concept does improve

Table 2
Results for 30 variable problem.

	C4.5			Simple GA w/o feature selection (GA)			GA with simultaneous FS (GAFS)			Hausdorff Dist. FS applied to GA (HS + GA)			Combining both FS techniques (HS + GAFS)		
Data set	(%)	Sec.	Size	(%)	Sec.	Size	(%)	Sec.	Size	(%)	Sec.	Size	(%)	Sec.	Size
1	82.4	56.33	55	83.77	96.46	4	81.84	43.67	4	80.88	74.81	3.62	78.42	41.27	2.4
2	83.3	89.28	47	82.63	92.74	4	85	48.88	4.8	84.39	75.4	3.2	86.05	47.02	4.4
3	82.8	58.68	47	80.53	100.42	3.8	80.44	38.47	3.6	82.37	79.5	4.4	81.84	54.24	5.2
4	84.2	40.84	33	79.56	86.88	3.6	78.69	42.46	3.8	82.37	69.29	2.8	83.07	38.46	2.8
5	85.6	46.14	39	83.77	87.17	3.6	77.72	46.18	4.6	85.44	82.72	4.2	82.28	39.89	3.4
6	85	47.96	31	83.51	70.32	5	80.62	44.35	4	83.33	51.31	4	84.04	41.21	3.6
7	77.2	81.28	51	87.11	85.79	4.8	86.41	45.04	4.6	83.33	53.65	4	84.48	38.58	3.2
8	84.6	59.85	39	82.72	67.54	4.2	80.79	44.37	3.8	85.09	52.71	3.2	83.6	42.53	3.6
9	80.2	62.08	31	81.23	74.14	4.4	78.86	42.03	3.6	80.61	56.35	3	81.23	50.75	3.6
10	78.5	66.51	43	78.95	60.91	3.6	77.72	34.97	2.8	85.35	52.37	3.8	83.07	43.73	4.2
Avg.	**82.38**	**60.90**	**41.6**	**82.38**	**82.24**	**4.1**	**80.81**	**43.04**	**3.96**	**83.32**	**64.81**	**3.62**	**82.81**	**43.77**	**3.64**
Std.	*2.85*	*15.16*	*8.44*	*2.41*	*13.25*	*0.5*	*2.94*	*3.93*	*0.6*	*1.76*	*12.69*	*0.55*	*2.07*	*5.31*	*0.8*
							Paired *t*-Test (2 tail)								
$P_{C4.5}$				n.s.	0.0068	2E-7	n.s.	0.004	2E-7	n.s.	n.s.	2E-7	n.s.	0.005	2E-7
P_{GA}							0.05	3.9E-6	n.s.	n.s.	4.2E-5	0.06	n.s.	7.5E-6	n.s.
P_{GAFS}										0.05	3.5E-4	n.s.	0.05	n.s.	n.s.
P_{HS+GA}													n.s.	5.5E-4	n.s.

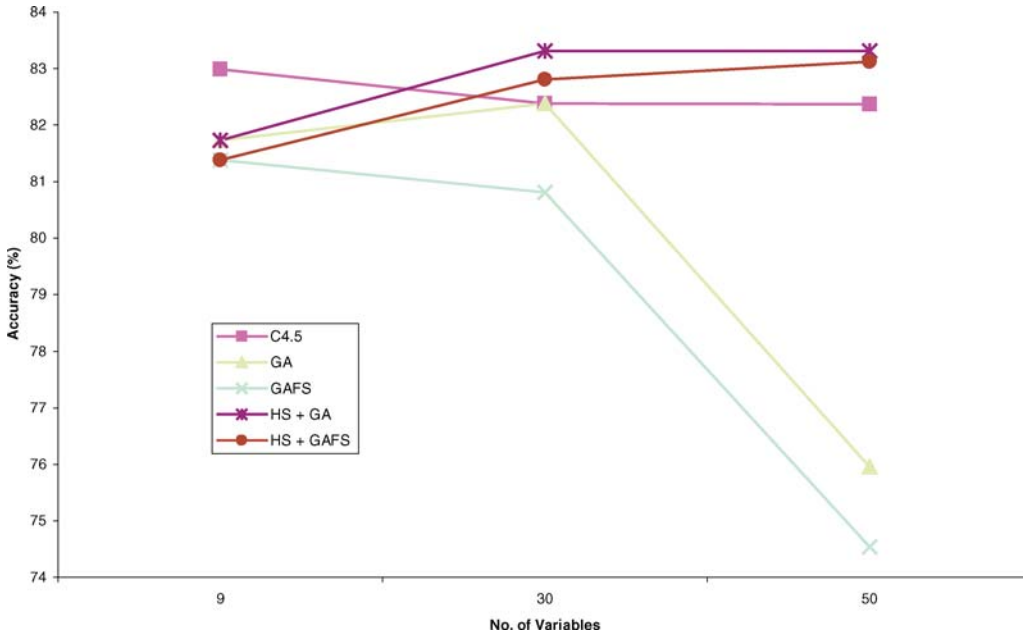

Figure 1. Comparative performance based on accuracy %.

Table 3
Results for 50 variable problem.

Data set	C4.5			Simple GA w/o feature selection (GA)			GA with simultaneous FS (GAFS)			Hausdorff Dist. FS applied to GA (HS + GA)			Combining both FS techniques (HS + GAFS)		
	(%)	Sec.	Size	(%)	Sec.	Size	(%)	Sec.	Size	(%)	Sec.	Size	(%)	Sec.	Size
1	82.1	160.43	37	72.9	126.18	4.6	76.32	94.21	4.8	84.04	28.38	3.8	83.33	25.92	4
2	82.1	146.73	37	75.09	150.77	5.8	73.42	100.81	5.8	82.81	66.05	3.8	84.21	50.61	3.6
3	83	111.41	45	79.12	161.99	6.8	76.67	87.18	5.0	82.02	62.74	3.8	83.51	52.30	4
4	85.2	101.6	39	75.26	145.42	5.8	74.12	109.02	6.2	84.04	60.63	3.6	82.37	41.12	2.8
5	78.1	106.61	47	79.04	171.63	7.2	71.14	91.80	5.2	84.65	68.61	3.8	83.68	21.17	4
6	83.4	102.47	35	77.72	153.59	6.4	73.86	80.52	4.6	84.91	62.82	3.8	84.65	52.12	4
7	86.1	181.86	41	76.31	132.74	4.8	78.25	91.67	5.0	85.18	71.49	3.8	84.91	49.38	3.4
8	83	221.51	33	72.02	130.60	4.8	73.25	102.58	6.0	83.69	71.37	4.4	81.32	49.05	3.8
9	77.3	155.5	39	74.47	155.9	6.2	73.07	83.78	4.8	81.14	29.94	4.8	82.02	24.54	4.8
10	83.4	213.4	39	77.55	171.14	7.6	75.35	86.97	5.0	80.61	60.99	3	81.32	48.79	3.6
Avg.	**82.37**	**150.15**	**39.2**	**75.95**	**150.0**	**6.0**	**74.54**	**92.86**	**5.24**	**83.31**	**58.3**	**3.86**	**83.13**	**41.5**	**3.8**
Std.	*2.77*	*45.06*	*4.26*	*2.44*	*16.23*	*1.04*	*2.09*	*8.97*	*0.56*	*1.60*	*15.86*	*0.47*	*1.31*	*12.6*	*0.52*
							Paired t-test (2 tail)								
$P_{C4.5}$				4E–4	n.s.	6E–9	7.6E–7	0.003	2E–9	n.s.	1.8E–4	1E–9	n.s.	2.9E–5	8E–9
P_{GA}							n.s.	1.6E–5	0.097	2.6E–5	1.9E–7	6E–4	3.7E–6	5.8E–8	1E–4
P_{GAFS}										2.1E–5	8.2E–5	2E–4	5.2E–7	2E–6	0.001
P_{HS+GA}													n.s.	0.002	n.s.

Table 4
Summary of results.

No. of Variables	C4.5			Simple GA w/o feature selection (GA)			GA with simultaneous FS (GAFS)			Hausdorff Dist. FS applied to GA (HS + GA)			Combining both FS techniques (HS + GAFS)		
	(%)	Sec.	Size	(%)	Sec.	Size	(%)	Sec.	Size	(%)	Sec.	Size	(%)	Sec.	Size
9	82.99	14.91	49.6	81.73	22.24	3.8	81.38	10.08	3.6	81.73	35.99	3.8	81.38	23.84	3.6
30	82.38	60.90	41.6	82.38	82.24	4.1	80.81	43.04	3.96	83.32	64.81	3.62	82.81	43.77	3.64
50	82.37	150.15	39.2	75.95	150.0	6.0	74.54	92.86	5.24	83.31	58.3	3.86	83.13	41.5	3.8

significantly without significant deterioration in the prediction accuracy of the concepts learned. However, the general performance trend as the problem size increases remains the same, with a steep degradation in performance as the problem size increases from 30 variables to 50 variables. As hypothesized earlier, the main advantage of the self-adaptive feature selection method would be to improve the computational time required, but it is interesting that in doing so it also generates a much more simpler concept.

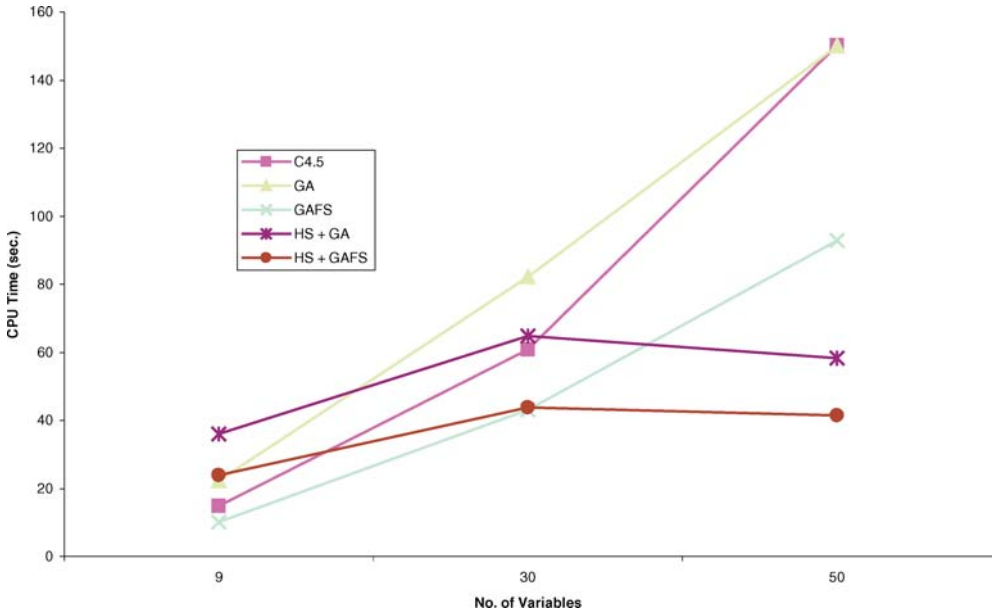

Figure 2. Comparative performance based on CPU time.

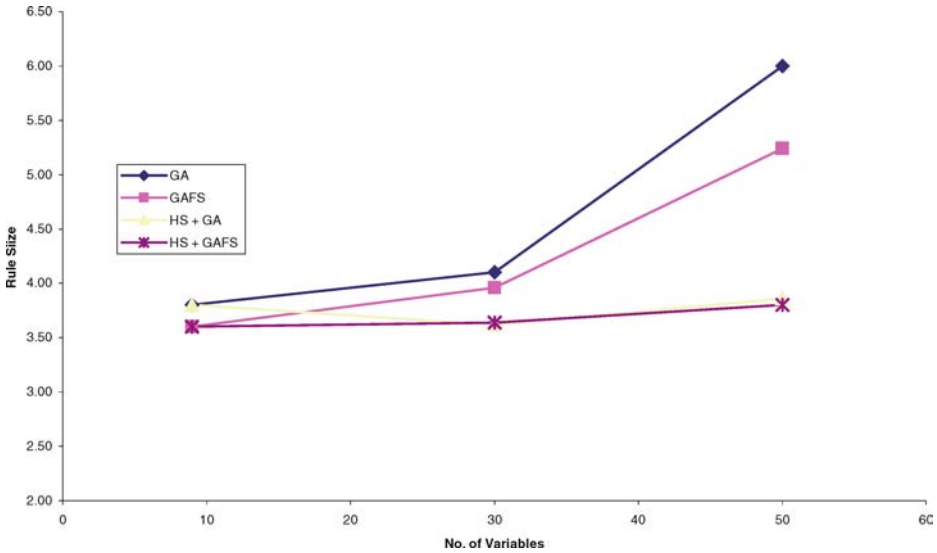

Figure 3. Comparative performance based on rule size.

- The wrapper approach of feature selection using Hausdorff distance (HS + GA) provides the most benefit on all the three criteria only for the 50 variable problem. Since the learning algorithm (GA in our case) has to be applied multiple times in a wrapper approach, this method of feature selection provides maximum advantage only

at larger problem sizes. The trade-off between the two feature selection approaches, GAFS and HS + GA, is evident in the 30-variable problem when the wrapper approach significantly improves the prediction accuracy but at the same time significantly adding to the computation time.

- It would seem that combining both of these approaches of feature selection might provide the best of both worlds—providing significant improvement in the prediction accuracy of the concepts at the same time tempering the computational requirement of the wrapper approach. That is indeed the case. For larger problem sizes the combined method significantly reduces the computation time compared to the wrapper approach without significantly affecting the prediction accuracy or the simplicity of the concept learned. For example, for the 30 variable and 50 variable problems the HS + GAFS method effectively reduces the computational time by about 33 and 29% respectively compared to the HS + GA method without a significant reduction in the accuracy of the learned classifiers. Note that the HS + GAFS method also results in significant improvement in the computation time and the complexity of the concept as compared to the C4.5 for larger problem sizes.
- The HS + GA and the HS + GAFS methods also provide a much more stable performance as the problem size increases, without drastically increasing the computational requirement.

7. Conclusions

Evolutionary algorithms are excellent for performing global search, especially when the search landscape is complex with multiple global as well local optima. They have been shown to be robust in searching for global optima without getting trapped in local optimum solutions. However, one of the reasons why the use of evolutionary algorithms (especially GAs) has not received widespread attention in the data mining community is that the GAs tend to be computationally expensive, and become prohibitively so, once the problem size increases. The only way of mining the robustness and adaptiveness of these techniques without sacrificing their ability to do a global search for data mining problems is to synergistically marry them to techniques that can provide faster, reliable, and parallel methods of extracting promising features that they can exploit.

We presented two such approaches of using feature selection in cooperation with a GA and showed how combining them can significantly improve the performance of the simple GA without a corresponding increase in the computational requirement even when the problem size increases.

Note

1. The 1-norm between two points $A(x_1, y_1)$ and $B(x_2, y_2)$ is defined as $d(A, B) = |x_1 - x_2| + |y_1 - y_2|$.

References

[1] A. Ahmed and Deriche Mohamed, An optimal feature selection technique using the concept of mutual information, in: *Proceedings of the International Symposium on Signal Processing and its Applications (ISSPA)* (2001) 477–480.

[2] H. Alt, B. Behrends and J. Bloemer, Approximate matching of polygonal shapes (extended abstract), in: *Proceedings of the Seventh Annual Symposium on Computational Geometry* (1991) 186–193.

[3] P.J. Angeline, Adaptive and self-adaptive evolutionary computations, in, *Computational Intelligence: A Dynamic Systems Perspectives*, M. Palaniswami, Y.Attikiouzel, R. Marks, D. Fogel, and T. Fukuda (eds.), IEEE Press. Piscataway, NJ, 1995, pp. 152–163.

[4] P.J. Angeline, Two self-adaptive crossover operations for genetic programming, in P. Angeline and K. Kinnear (eds.), *Advances in Genetic Programming II*, MIT Press, Cambridge, MA, 1996, pp. 152–163.

[5] D. Aranjo, H. Lopes and A, Freitas, Rule discovery with a parallel genetic algorithm, in *Data Mining with Evolutionary Algorithms* (2000) 89–94.

[6] C. Babcock, Parallel processing mines retail data, Computer World 6 (1994).

[7] Thomas Back, Self-adaptation in genetic algorithms, in F.J. Varela and P. Bourgine (eds.), *Towards a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, MIT Press. Cambridge, MA, pp. 263–271.

[8] J. Bala, K. De Jong, J. Huang, H. Vafaei and H. Wechsler, Hybrid learning using genetic algorithms and decision trees for pattern classification, in *Proc. of 14th Intl. Joint Conf. on Artificial Intelligence* (IJCAI), (1995).

[9] J. Bala, K. De Jong, and P. Pachowicz, Multistrategy learning from engineering data by integrating inductive generalization and genetic algorithms, in *Machine Learning: A Multistartegy Approach Volume IV*, R. Michalski and G. Tecuci, (eds.), 1994, San Francisco: Morgan Kaufmann.

[10] R. Battiti, Using mutual information for selecting features in supervised neural net learning, IEEE Transactions on Neural Networks 5(4) (1994) 537–550.

[11] J.M. Benitez, J.L. Castro, C.J. Mantas, and F. Rojas, A Neuro-Fuzzy Approach for Feature Selection, Proceedings of IFSA World Congress and 20th NAFIPS International Conference 2 (2001) 1003–1008.

[12] S, Bhattacharya, Evolutionary algorithms in data mining: Multiobjective performance modeling for direct marketing, in: *Proc. of 6 th ACM SIGKDD International Conf. On Knowledge Discovery and Data Mining*, (2000) 465–473.

[13] P.S. Bradley, O.L. Mangasarian and W.N. Street, Feature selection in mathematical programming, INFORMS Journal on Computing 10(2) (1998).

[14] L. Breiman, J. Friedman, R. Olshen and C. Stone, *Classification and Regression Trees* Belmont, Calif.: Wadsworth 1984).

[15] Bjorn Chambless and David Scarborough, Information theoretic feature selection for a neural behavioral model, Proceedings of the International Joint Conference on Neural Networks (IJCNN-01) 2 (2001) 1443–1448.

[16] K. Chellapilla and D.B. Fogel, Exploring self-adaptive methods to improve the efficiency of generating approximate solutions to travelling salesman problems using evolutionary programming, in P.J. Angeline, R.G. Reynolds, J.R. McDonnell and R. Eberhart, (eds.) *Evolutionary Programming VI*, Springer, 1997).

[17] Coetzee, Frans, M., Eric Glover, Steve Lawrence and C. Lee Giles, Feature Selection in Web Applications by ROC Inflections and Powerset Pruning, in *Proceedings of the Symposium on Applications and the Internet* (2001), 5–14.

[18] T.M. Cover, The best two independent measurements are not the two best, IEEE Transactions on Systems, Man, and Cybernetics, SMC-4:1 (1974) 116–117.

[19] A. Csaszar, *General Topology* Adam Hilger, Bristol: 1978.

[20] B. Dasarathy, *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*, (IEEE Computer Society Press. Los Alamitos, CA., 1991).

[21] S. Dzeroski, Inductive logic programming and knowledge discovery in databases, in *Advances in Knowledge Discovery and Data Mining*, AAAI Press (Menlo Park, Calif., 1996) pp. 117–152.
[22] J.D. Elashoff, R.M. Elashoff and G.E. Goldman, On the choice of variables in classification problems with dichotomous variables, Biometrika 54 (1967) 668–670.
[23] J. Elder and D. Pregibon, A Statistical perspective on knowledge discovery aaai press (menlo park, calif., in databases, in *Advances in Knowledge Discovery and Data Mining*, 1996).
[24] T. Elomaa and E. Ukkonen, A geometric approach to feature selection, in *Proceedings of the European Conference on Machine Learning* (1994) 351–354.
[25] C. Emmanouilidis, A. Hunter and J. MacIntyre, A multiobjective evolutionary setting for feature selection and a commonality-based crossover operator, in: *Proc. of Congress on Evolutionary Computation* (2000) 309–316.
[26] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth and R. Uthurusamy, *Advances in Knowledge Discovery and Data Mining*, AAAI Press, Menlo Park, CA, (1996).
[27] D.B. Fogel, L.J. Fogel and J.W. Atmar, Meta-evolutionary programming, in R.R. Chen (ed.). *Proceedings of 25th Asilomar Conference on Signals, Systems, and Computers* (1991), pp. 540–545.
[28] J. Friedman, Multivariate adaptive regression splines, Annals of Statistics 19 (1989, 1992) 1–141.
[29] D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning* (1989). Reading, MA: Addison-Wesley Publishing Co., Inc.
[30] C. Guerra-Salcedo and D. Whitley, Genetic approach to feature selection for ensemble creation, in *Proc. of the Genetic and Evolutionary Computation Conference* (1999) 236–243.
[31] J. Holland, *Adaptations in Natural and Artificial Systems*. 2nd Ed., (1992). MIT Press.
[32] W. Hsu, M. Welge, T. Redman and D. Clutter, Genetic wrappers for constructive induction in high-performance data mining, in: *Proc. of the Genetic and Evolutionary Computation Conference* (2000) 765.
[33] H. Ishibuchi and T. Nakashima, Multi-objective pattern and feature selection by a genetic algorithm, in: *Proc. of the Genetic and Evolutionary Computation Conference* (2000) 1069–1076.
[34] G. John, R. Kohavi and K. Pfleger, Irrelevant features and the subset selection problem, in: *Proceedings of the 11th International Conference on Machine Learning* (1994) 121–129.
[35] J. Kolodner, *Case-Based Reasoning*. San Francisco: Morgan Kaufmann (1993).
[36] Y. Kim, W. Street and F. Menczer, Feature selection in unsupervised learning via evolutionary search, in: *Proc. of the 6th ACM SIGKDD Intl. Conf. On Knowledge Discovery and Data Mining* (2000) 365–369.
[37] K. Kira and L.A. Rendell, A practical approach to feature selection, in: *Proc. of the 9th International Conference on Machine Learning* (1992) 249–256.
[38] J. Kittler, Mathematical methods of feature selection in pattern recognition, International Journal of Man-Machine Studies 7 (1975) 609–637.
[39] P. Lanzi, Fast feature selection with genetic algorithms: A filter approach, in: *Proc. of IEEE Intl. Conf. on Evolutionary Computation* (1997) 537–540.
[40] I. Lee, R. Sikora and M. Shaw, A genetic algorithm based approach to flexible flow-line scheduling with variable lot sizes, IEEE Transactions on Systems, Man, and Cybernetics 27B(1) (1995) 36–54.
[41] J. Levenick, Inserting Introns Improves Genetic Algorithm Success Rate: Taking a Cue from Biology, in R. Belew and L. Booker (eds.) *Proc. of the Fourth Intl. Conf. on Genetic Algorithms*, (1991) pp. 123–127.
[42] H. Liu and R. Setiono, Feature selection via discretization, IEEE Transactions on Knowledge and Data Engineering 9(4) (1997) 642–645.
[43] S.B. Nadler jr, *Hyperspaces of Sets* (Marcel Dekker, New York: 1978).
[44] E. Noda, A. Freitas and H. Lopes, Comparing a genetic algorithm with a rule induction algorithm in the data mining task of dependence modeling, in: *Proc. of the Genetic and Evolutionary Computation Conference* (2000) 1080.

[45] P. Nordin, F. Francone and W. Banzhaf, Explicitly defined introns and destructive crossover in genetic programming, in, P. Angeline and K. Kinnear (eds.), *Advances in Genetic Programming: Volume 2*, (1996), 111–134.
[46] D. Opitz, An evolutionary approach to feature set selection, in: *Proc. of the Genetic and Evolutionary Computation Conference*, (1999), 803.
[47] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*. (Morgan Kaufmann, San Francisco, 1988).
[48] W. Punch, E. Goodman, M. Pei, L. Chia-Shun, P. Hovland and R. Enbody, Further research on feature selection and classification using genetic algorithms, in S. Forrest (ed.), *Proceedings of the 5th International Conference on Genetic Algorithms*, (1993) 557–564.
[49] J. Quinlan, *C4.5: Programs for Machine Learning*. (Morgan Kaufmann, San Francisco, 1992).
[50] M. Raymer, W. Punch, E. Goodman, Sanschagrin and L, Kuhn, Simultaneous feature scaling and selection using a genetic algorithm, in *Proc. of the 7th Intl. Conf. On Genetic Algorithms* (1997) 561–567.
[51] H.P. Schwefel, *Numerical Optimization of Computer Models*. (Wiley, Chichester, 1981).
[52] R. Sikora, Learning control strategies for a chemical process: A distributed approach, IEEE Expert, (1992) 35–43.
[53] R. Sikora and M. Shaw, A double-layered learning approach to acquiring rules for classification: Integrating genetic algorithms with similarity-based learning, ORSA Journal on Computing 6(2) (1994) 174–187.
[54] R. Sikora and S. Piramuthu, An intelligent fault diagnosis system for robotic machines, International Journal of Computational Intelligence and Organizations 1(3) (1996) 144–153.
[55] G.T. Toussaint, Note on optimal selection of independent binary-valued features for pattern recognition, IEEE Transactions on Information Theory IT-17 (1971), 618.
[56] P. Turney, How to shift bias: Lessons from the baldwin effect, Evolutionary Computation 4(3) (1997) 271–295.
[57] H. Vafaie and K. De Jong, Improving a rule induction system using genetic algorithms, in R. Michalski and G. Tecuci (eds.), *Machine Learning: A Multistartegy Approach Volume IV*, (San Francisco: Morgan Kaufmann 1994).
[58] J. Whittaker, *Graphical Models in Applied Multivariate Statistics*. (Wiley, New York).
[59] J. Yang and V. Honavar, Feature subset selection using a genetic algorithm, IEEE Intelligent Systems 13(2) (1998) 44–49.