

Predictive and comprehensible rule discovery using a multi-objective genetic algorithm

S. Dehuri^a, R. Mall^{b,*}

^a P.G. Department of Information and Communication Technology, Fakir Mohan University, Vyasa Vihar, Balasore 756019, India

^b Department of Computer Science and Engineering, Indian Institute of Technology, Kharagpur 721302, India

Received 21 January 2005; accepted 23 March 2006

Available online 23 June 2006

Abstract

We present a multi-objective genetic algorithm for mining highly predictive and comprehensible classification rules from large databases. We emphasize predictive accuracy and comprehensibility of the rules. However, accuracy and comprehensibility of the rules often conflict with each other. This makes it an optimization problem that is very difficult to solve efficiently. We have proposed a multi-objective evolutionary algorithm called improved niched Pareto genetic algorithm (INPGA) for this purpose. We have compared the rule generation by INPGA with that by simple genetic algorithm (SGA) and basic niched Pareto genetic algorithm (NPGA). The experimental result confirms that our rule generation has a clear edge over SGA and NPGA.

© 2006 Elsevier B.V. All rights reserved.

Keywords: Simple genetic algorithm; Pareto optimal solutions; Niched Pareto genetic algorithm; Data mining

1. Introduction

The commercial and research interests in mining for classification rules are increasing rapidly, because the amount of data being generated and stored in databases of organizations is already enormous and continues to grow very fast. This large amount of stored data normally contains valuable hidden knowledge, which if harnessed could be used to improve the decision-making process of an organization. For instance, data about previous sales might contain interesting relationships between products, customer segmentation and buying habits of customers. The discovery of such relationships can be very useful to efficiently manage the sales of a company. However, the volume of the archival data often exceeds several gigabytes and sometimes even terabytes, such an enormous volume of data is beyond the manual analysis capability of human beings. Thus, there is a clear need for developing automatic

methods for extracting knowledge from data that not only have a high predictive accuracy but also are comprehensible by users [1–3]. The user should be able to understand the mining system's results and combine them with his/her knowledge to make a well-informed decision, rather than blindly trusting the incomprehensible output of a “black box” system.

Evolutionary algorithms (EAs) have inspired many research efforts for optimization as well as rule generation [4,5]. Traditional rule generation methods, are usually accurate, but have brittle operations. Evolutionary algorithms on the other hand provide a robust and efficient approach to explore large search space. One of the EAs called simple genetic algorithm (SGA) introduced by J.H. Holland (1975) [4] and further extension can found in [6,7] is good for rule generation satisfying a single objective. However, practical rule generation is naturally posed as multi-objective problems with two criteria: (i) predictive accuracy and (ii) comprehensibility [3]. The SGA normally handles problems with such criteria by converting them into a single objective problem. The single objective is formed from a linear combination of the multiple objective

* Corresponding author.

E-mail addresses: satchi_d@silicon.ac.in (S. Dehuri), rajib@cse.iitkgp.ernet.in (R. Mall).

functions. However, this approach is unsatisfactory due to the nature of the optimality conditions for multiple objectives. In the presence of multiple and conflicting objectives, the resulting optimization problem gives rise to a set of optimal solutions, instead of just one optimal solution. Multiple optimal solutions exist because no single solution can be a substitute for multiple conflicting objectives. In order to overcome this difficulty we have proposed the multi-objective genetic algorithm called INPGA an improved version of NPGA [8,9] for rule generation.

We propose to use INPGA to discover high-level prediction rules of the form:

IF some conditions hold on the values of a set of predicting attributes
 THEN predict a value for the goal attribute.

In other words, the value of a special attribute called the goal attribute is predicted by the values given for other attributes called the predicting attributes.

The INPGA rule generation is to associate each individual of the population with the same predicted class, which is never modified during the running of the algorithm. We would need to run the INPGA at least for the specified number of classes. So that in the i th run, the algorithm discovers only rules i th class [10]. We shall use the results reported in [11] for comparison with our results. By comparison it has been observed that the predictive accuracy and comprehensibility is encouraging in INPGA over to SGA and NPGA.

This paper is organized as follows. Section 2 discusses the SGA for classification rule generation. In Section 3, we describe evolutionary algorithms for multi-objective problems. In Section 4, we have discussed the improved niched Pareto genetic algorithm. The implementation of our simulation experiments is discussed in Section 5. Finally, Section 6 concludes this paper.

2. Using SGA for classification rule generation

In this section, we review the function of SGA for rule generation. Genetic algorithms are probabilistic search algorithms characterized by the fact that a number N of potential solutions (called individuals $I_k \in \Omega$, where Ω represents the space of all possible individuals) of the optimization problem simultaneously sample the search space. This population $P = \{I_1, I_2, \dots, I_N\}$ is modified according to the natural evolutionary process: after initialization, selection $S: I^N \rightarrow I^N$ and recombination $\pi: I^N \rightarrow I^N$ are executed in a loop until some termination criterion is reached. Each run of the loop is called a generation and $P(t)$ denotes the population at generation t .

The selection operator is intended to improve the average quality of the population by giving individuals of higher quality a higher probability to be copied into the next generation. Selection thereby focuses on the search of promising regions in the search space. The quality of an individual is measured by a fitness function $f: P \rightarrow R$.

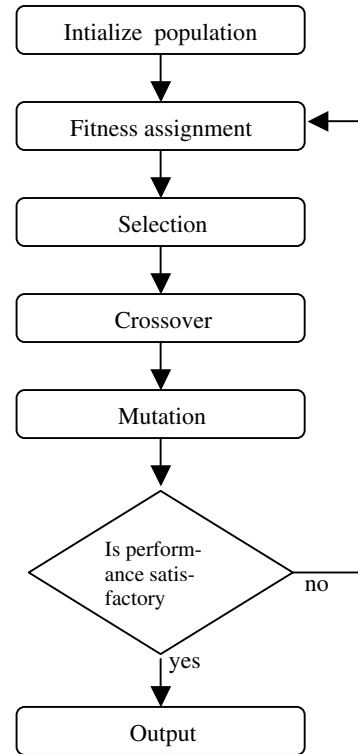


Fig. 1. Flow diagram of SGA.

Recombination changes the genetic material in the population either by crossover or by mutation in order to obtain new points in the search space. Fig. 1 depicts the steps that are performed in SGA.

The following subsection discusses the individual representations, fitness function, and genetic operators used for classification rule discovery.

2.1. Genetic representations

Each individual in the population represents a candidate rule ‘ \mathcal{R} ’ of the form ‘if A then C ’. The antecedent of this rule can be formed by a conjunction of at most $n - 1$ attributes, where n is the number of attributes being mined. Each condition is of the form $A_i = V_{ij}$, where A_i is the i th attribute and V_{ij} is the j th value of the i th attribute’s domain. The consequent consists of a single condition of the form $G_k = V_{kl}$, where G_k is the k th goal attribute and V_{kl} is the l th value of the k th goal attribute’s domain. The user specifies the goal attribute that is of interest to him.

A string of fixed size encodes an individual with n genes representing the values that each attribute can assume in the rule. This encoding is shown in Fig. 2. The algorithm automatically chooses the best goal attri-

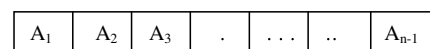


Fig. 2. Chromosome representation.

bute to put in the consequent, for a given rule antecedent. If an attribute is not present in the rule antecedent, the corresponding value in gene is “−1”. This value is a flag to indicate that the attribute does not occur in the rule antecedent. Hence, this encoding effectively represents a variable-length individual (rule).

2.2. Fitness function

As discussed in Section 1, the discovered rules should have (a) high predictive accuracy and (b) high comprehensibility. In this subsection, we discuss how these multiple criteria can be incorporated into a single objective fitness function.

2.2.1. Comprehensibility metric

There are various ways to quantitatively measure rule comprehensibility. The standard way of measuring comprehensibility is to count the number of rules and the number of conditions in these rules. If these numbers increase then the comprehensibility decreases.

If a rule can have at most ‘ M_c ’ conditions, the comprehensibility ‘ ζ ’ of a rule ‘ \mathfrak{R} ’ can be defined as:

$$\zeta(\mathfrak{R}) = 1 - (N_c(\mathfrak{R})/M_c),$$

where ‘ $N_c(\mathfrak{R})$ ’ is the number of conditions in the rule \mathfrak{R} .

2.2.2. Predictive accuracy

As already mentioned, our rules are of the form IF $A_1 \wedge A_2$ THEN C . The antecedent part of the rule is a conjunction of conditions. A very simple way to measure the predictive accuracy of a rule ‘ $P(\mathfrak{R})$ ’ is

$$P(\mathfrak{R}) = \frac{|A\&C|}{|A|}, \quad (1)$$

where $|A|$ is the number of examples satisfying all the conditions in the antecedent A and $|A\&C|$ is the number of examples that satisfy both the antecedent A and the consequent C . Intuitively, this metric measures predictive accuracy in terms of how many cases both antecedent and consequent hold out of all cases where the antecedent holds.

A variation of (1) is :
$$P(\mathfrak{R}) = \frac{(|A\&C| - 1/2)}{A}, \quad (2)$$

where $|A\&C|$ is the number of examples that satisfy both the rule antecedent and the consequent. The term $1/2$ is subtracted in the numerator of Eq. (1) to penalize rules covering few training examples.

The fitness function is computed as the arithmetic weighted mean of comprehensibility and predictive accuracy. Finally, the fitness function is given by:

$$f(x) = \frac{w_1 \times \zeta(\mathfrak{R}) + w_2 \times P(\mathfrak{R})}{w_1 + w_2}, \quad (3)$$

where w_1 and w_2 are user-defined weights.

2.3. Genetic operators

The crossover operator we consider is based on uniform crossover [12,13]. There is a probability for applying

crossover to a pair of individuals and another probability for swapping each gene (attribute)’s value in the genome (rule antecedent) of two individuals. After crossover is complete, the algorithm analyses if any invalid individual was created. If so, a repair operator is used to produce valid-genotype individuals. The mutation operator randomly transforms the value of an attribute into another value belonging to the same domain of the attribute.

Besides crossover and mutation, the insert and remove operators directly try to control the size of the rules being evolved. Thereby thus influence the comprehensibility of the rules. These two operators randomly insert and remove, respectively, a condition in the rule antecedent. These operators are not part of the regular GA. However, we have introduced them here for suitability in our rule generation scheme.

3. Evolutionary algorithms for multi-objective problems

There are many multi-objective problems requiring simultaneous optimization of several competing objectives. Formally it can be stated as follows: We want to find $\vec{x} = (x_1, x_2, \dots, x_n)$ which maximizes the values of ‘ p ’ objective functions $F(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), \dots, f_p(\vec{x}))$ within a feasible domain Ω . Generally, the answer is not a single solution but a family of solutions called a *Pareto-optimal set*.

Definitions:

- (I) A vector $\vec{u} = (u_1, u_2, \dots, u_p)$ is said to dominate $\vec{v} = (v_1, v_2, \dots, v_p)$ iff \vec{u} is partially greater than \vec{v} , i.e., $\forall i \in \{1, 2, 3, \dots, p\}, u_i \geq v_i \wedge \exists i \in \{1, 2, \dots, p\}: u_i > v_i$.
- (II) A solution $x \in \Omega$ is said to be Pareto-optimal with respect to Ω iff there is no $x' \in \Omega$ for which $\vec{v} = F(x') = (f_1(x'), f_2(x'), \dots, f_p(x'))$ dominates $\vec{u} = F(x) = (f_1(x), f_2(x), \dots, f_p(x))$.
- (III) For a given multi-objective problem $F(x)$, the Pareto-optimal set P_S is defined as:
$$P_S = \{x \in \Omega | \neg \exists x' \in \Omega : F(x') \geq F(x)\}.$$
- (IV) For a given multi-objective problem $F(x)$ and Pareto optimal set P_S , the Pareto front P_f is defined as:
$$P_f = \{\vec{u} = F(x) = (f_1(x), f_2(x), \dots, f_p(x)) | x \in P_S\}.$$

Optimization methods generally try to find a given number of Pareto-optimal solutions which are uniformly distributed in the Pareto-optimal set, such solutions provide the decision maker sufficient insight into the problem to make the final decision. Methods such as weighted sum, ϵ -constraint, and goal programming have been proposed to search for Pareto optima [14,15]. However, an a priori articulation of the preferences to the objectives is required, which is often hard to decide beforehand. Besides, these

methods can only find one solution at a time. Other solutions cannot be obtained without re-computation with the free parameters reset.

By contrast, genetic algorithms (GAs) [7] maintain a population and thus can search for many non-dominated solutions in parallel. GA's ability to find a diverse set of solutions in a single run and its exemption from demand for objective preference information renders it immediate advantage over aforementioned techniques. A lot of multi-objective GAs (MOGAs) [16,17] have been proposed. Basically, an MOGA is characterized by its fitness assignment and diversity maintenance strategy.

In fitness assignment, most MOGAs fall into two categories, non-Pareto and Pareto-based. Non-Pareto methods use the objective values as the fitness value to decide an individual's survival. Schaffer's VEGA is such a method. The more recent predator-prey approach [18] is another one, where some randomly walking predators will kill a prey or let it survive according to the prey's value in one objective. In contrast, Pareto-based methods measure individuals fitness according to their dominance property. The non-dominated individuals in the population are regarded as fittest regardless of their single objective values. Since Pareto-based approaches respect better the dominance nature of multi-objective problems, their performance is reported to be better.

Diversity maintenance strategy is another characteristic of MOGAs. It works by keeping the solutions uniformly distributed in the Pareto-optimal set, instead of gathering solutions in a small region only. Fitness sharing [19], which reduces the fitness of an individual if there are some other candidates nearby, is one of the most renowned techniques. Restricted mating, where mating is permitted only when the distance between two parents is large enough, is another technique. More recently, some parameter-free techniques were suggested. The techniques used in SPEA [20] and NSGA-II [21] are two examples of such techniques. PAES [22], SPEA [20], and NSGA-II [21] are representatives of current MOGAs. They all adopt Pareto-based fitness assignment strategy and implement elitism, an experimentally verified technique known to enhance performance.

In this paper a new improved niched Pareto genetic algorithm has been proposed and is discussed in Section 5. Its performance in rule generation has been compared with the niched Pareto genetic algorithm and simple genetic algorithm.

4. Our proposed niched Pareto genetic algorithm

4.1. The basic niched Pareto GA

The most widely implemented selection technique for GAs is tournament selection. However, tournament selection assumes that we want a single answer to the problem. After a certain number of generations the population will converge to a uniform one. To avoid convergence and

maintain multiple Pareto optimal solutions, the tournament selection is altered in two ways. First, Pareto domination tournament is introduced. Second, when a non-dominant tournament (i.e., a tie), sharing is implemented to determine the winner.

4.1.1. Pareto domination tournaments

The binary relation of domination leads naturally to a binary tournament in which two randomly selected individuals are compared. If one dominates the other, it wins. Initially, such a small local domination criterion is used, but soon found that it produced insufficient domination pressure. There were too many dominated individuals in later generations. It seemed that a sample size of two was too small to estimate an individual's true domination ranking.

Because we wanted more domination pressure, and more control of that pressure, a sampling scheme is implemented, as follows. Two candidates for selection are picked at random from the population. A comparison set of individuals is also picked randomly from the population. Each of the candidates is then compared against each individual in the comparison set. If one candidate is dominated by the comparison set, and the other is not, the later is selected for reproduction. If neither or both are dominated by the comparison set, then we must use sharing to choose a winner, as we explained later. The sample size t_{dom} (size of comparison set) gives us control over selection pressure, or what we call domination pressure. The performance of the niched Pareto GA is somewhat sensitive to the amount of domination versus sharing pressure applied [20].

A problem will arise if both candidates are on the current non-dominated front since neither will be dominated. Even if the front, a small t_{dom} could mean that neither appears dominated. And of course both could be dominated. How is a winner then chosen in such a tie? If we choose the winner at random, genetic drift will cause the population to converge to a single region of the Pareto front. To prevent this we implement a form of sharing when there is no preference between two individuals.

4.1.2. Sharing on the non-dominated frontier

Fitness sharing was introduced by Goldberg and Richardson, [19], and has been applied successfully to a number of difficult real-world problems. The goal of fitness sharing is to distribute the population over a number of different peaks in the search space, with each peak receiving a fraction of the population in proportion to the height of that peak.

To achieve this distribution, sharing calls for the degradation of an individual's objective fitness f_i by niche count m_i calculated for that individual. This degradation is obtained by simply dividing the objective fitness by the niche count to find the shared fitness: f_i/m_i . The niche count m_i is an estimate of how crowded is the neighborhood (niche) of individual i . It is calculated over all individuals in the current population: $m_i = \sum_{j \in Pop} Sh[d[i,j]]$, where $d[i,j]$ is the distance between individuals i and j and $Sh[d]$

is the sharing function. $Sh[d]$ is a decreasing function of $d[i, j]$, such that $Sh[0] = 1$ and $Sh[d \geq \sigma_{share}] = 0$. Typically, the triangular sharing function is used, where $Sh[d] = 1 - d/\sigma_{share}$ and $Sh[d] = 0$ for $d \geq \sigma_{share}$. Here, σ_{share} is the niche radius, fixed by the user at some estimate of the minimal separation desired or expected between the goal solutions. Individuals within σ_{share} distance of each other degrade each other's fitness, since they are in the same niche, but convergence of the full population is avoided. As one niche "fills up", its niche count increases to the point that its shared fitness is lower than that of other niches.

Fitness sharing was originally combined with fitness proportionate (e.g., roulette wheel) selection. When sharing is combined with the more popular tournament selection, however, the Niche GA exhibits chaotic behavior. The wild fluctuations in niche subpopulations induced by the "naïve" combination of sharing and tournament selection can be avoided. Oei, Goldberg, and Chang suggest the use of tournament selection with continuously updated sharing, in which niche counts are calculated not by using the current population, but rather the partly filled next generation population. This method was used successfully by Goldberg, Deb, and Horn on a Niching difficult problem. Also, it was found empirically that sampling the population was sufficient to estimate the niche count and so avoid the $O(N^2)$ comparisons needed to calculate exactly the m_i . We incorporate both techniques (continuously updated sharing and niche count sampling) in the Niche Pareto GA.

In any application of sharing, we cannot implement genotypic sharing, since we always have a genotype (the encoding). But Deb's work indicated that in general, phenotypic sharing is superior to genotypic sharing. Intuitively, we want to perform sharing in a space we care "more about", that is, some phenotypic space. Since we are interested in maintaining diversity along the phenotypic Pareto optimal front, which exists only in attribute space, it makes sense to perform our sharing in attribute space.

When the candidate solutions are either both dominated or both non-dominated, it is likely that they are in the same equivalence class (in the partial order induced by the domination relation). Because we are interested in maintaining diversity along the front, and most of the individuals in these equivalence classes can be labeled "equally" fit, we do not implement any form of fitness degradation according to the niche count. Instead, the "best fit" candidate is determined to be that candidate which has the least number of individuals in its niche and thus the smallest niche count. We call this type of *sharing equivalence class sharing*.

Fig. 3 illustrates how this form of sharing should work between two non-dominated individuals. Here, we are maximizing along the x -axis and minimizing along the y -axis. In this case, the two candidates for selection are not dominated by the comparison set; but are in the Pareto optimal subset (the dashed region) of the union of the comparison set and the candidates. From a Pareto point of view, neither candidate is preferred. But if we want to maintain use-

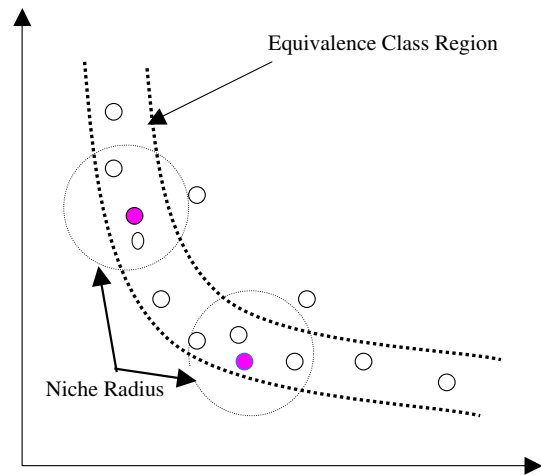


Fig. 3. Equivalence class sharing.

ful diversity (i.e., a representative sampling of the Pareto frontier), it is apparent that it would be best to choose the candidate that has the smaller niche count. In this case, it is candidate 2.

4.2. The improved niched Pareto GA

When the candidates are either both dominated or both non-dominated, then in NPGA the equivalence class sharing is adopted. But, that is not the only measure; we have to also consider the measure that can maintain useful diversity in the Pareto set. The following approach can be suitable to achieve both the goals, it is called improved niched Pareto genetic algorithm (INPGA).

- (I) Find out the center of gravity of both niche radius (μ_1 and μ_2) as:

$$\mu_1 = \left(\sum_{i \in \sigma_{share1}} x_i \right) / |\sigma_{share1}|$$

and

$$\mu_2 = \left(\sum_{i \in \sigma_{share2}} x_i \right) / |\sigma_{share2}|.$$

- (II) Calculate the standard deviation of each point of both radii.

$$\sigma_1 = \sqrt{\sum_{x_i \in \sigma_{share1}} (x_i - \mu_1)^2}$$

and

$$\sigma_2 = \sqrt{\sum_{x_j \in \sigma_{share2}} (x_j - \mu_2)^2}.$$

- (III) The candidate having larger SD is chosen.

Fig. 4 illustrates how to maintain diversity in the overall proposed method.

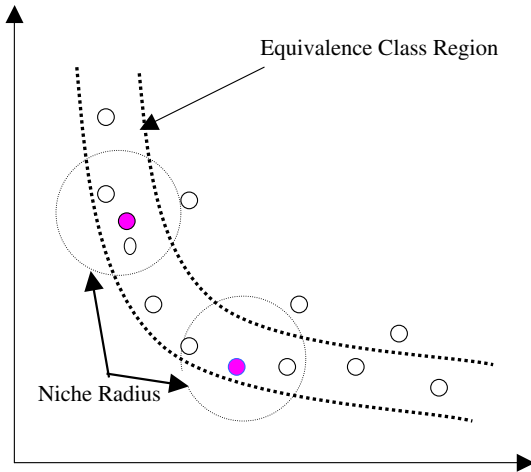


Fig. 4. Equivalence class sharing with tie breaking.

5. Simulation

5.1. Description of the dataset

The simulation was performed using the zoo and nursery dataset. The zoo and nursery dataset used to test the algorithm was obtained from the UCI machine repository (<http://www.ics.uci.edu/>). This data set is normally used as a benchmark for evaluating algorithms performing classification task.

5.1.1. Zoo data

The zoo dataset contains 101 instances and 18 attributes. Each instance corresponds to an animal. In the pre-processing phase the attribute containing the name of the animal was removed, since this attribute has no generalization power. The attributes in the zoo data set are all categorical. The attribute names in the data set are as follows: hair(h), feathers(f), eggs(e), milk(m), predator(p), toothed(t), domestic(d), backbone(b), fins(fs), legs(l), tail(tl), cat-size(c), airborne(a), aquatic(aq), breathes(br), venomous(v) and type(ty). Except type and legs, all other attributes are Boolean. The goal attributes are type 1–7. The type 1 has 41 records, type 2 has 20 records, type 3 has 5 records, type 4–7 has 13, 4, 8, and 10 records, respectively.

5.1.2. Nursery data

This dataset has 12,960 records and nine attributes, all of them categorical. The ninth attribute is treated as class

Table 1
Summary of nursery dataset

Attributes	Values
Parents	Usual, pretentious, great_pret
Has_nurs	Proper, less_proper, improper, critical, very_crit
Form	Complete, completed, incomplete, foster
Children	1,2,3, more
Housing	Convenient, less_conv, critical
Finance	Convenient, inconv
Social	Nonprob, slightly_prob, problematic
Health	Recommended, priority, not_recom

attribute and there are five classes: not_recom (NR), recommended (R), Very_recom (VR), Priority (P), and spec_prior (SP). The attributes and corresponding values are listed in Table 1.

The data-mining algorithm needs to discover rules by accessing the training set only. In order to do this, the algorithm has access to the values of both predicting attributes and the goal attribute of each example (record) in the training set. Once the training process is finished and the algorithm finds a set of classification rules, the predictive performance of these rules is evaluated on the test set, which was not seen during training.

Note that, once we take into account the large number of attributes, this can be considered a difficult classification problem.

5.2. Results

The experiments have been performed using MATLAB 5.3 on a Linux server. The data-specific parameters and the parameters, which are encountered during the rule discovery, are listed in Table 2.

For each of the dataset the simple genetic algorithm had 100 individuals in the population and was run for 500 generations. The parameters values such as P_c , P_m , R_m , and R_I were sufficient to find some good individuals. The following computational protocols are used in the basic niched Pareto genetic algorithm as well as the proposed improved niched Pareto genetic algorithm for rule generation. The data set is divided into two parts: training set and test set. Here, we have used 30% for training set and rest are test set. We represent the predicted class to all individuals of the population, which is never modified during the running of the algorithm. Hence, for each class we run the algorithms separately and get the corresponding rules. The generated rules of NPGA and INPGA have been compared with the SGA and all rules are listed in the following table.

Tables 3–5 show the result generated by SGA, NPGA and INPGA, respectively, from zoo dataset. The table has four columns namely class#, mined rules, predictive accuracy, and comprehensibility.

Tables 6–8 show the result generated by SGA, NPGA, and INPGA, respectively, from nursery dataset. The table has four columns namely class#, mined rules, predictive accuracy, and comprehensibility. Figs. 5–8 depict the comparative performance of the three approaches. From Figs. 5–8 it can be observed that in SGA the predictive accuracy is good but comprehensibility is very poor as compared to

Table 2
Parameters used for our simulation studies

Dataset	P	P_c	P_m	t_{size}	R_m	R_I	σ_{share}
Zoo	100	0.8	0.03	15	[0,0.7]	[0,0.8]	11
Nursery	500	0.75	0.002	50	[0.2,0.8]	[0,0.6]	20

P , population size; P_c , probability of crossover; P_m , probability of mutation; t_{size} , tournament size; R_m , removal operator; R_I , insert operator; σ_{share} , niche radius.

Table 3
Rules generated by SGA from zoo dataset

Class#	Mined rules	Predictive accuracy	Comprehensibility
1	If (hair = 1) \wedge (eggs = 0) \wedge (venomous = 0) \wedge (domestic = 0) Then (type = 1)	0.9545	0.8
2	If (hair = 0) \wedge (feathers = 1) \wedge (venomous = 0) \wedge (legs = 2) \wedge (domestic = 0) Then (type = 2)	1.0	0.7333
3	If (eggs = 1) \wedge (aquatic = 0) \wedge (predator = 1) (toothed = 1) \wedge (fins=0) \wedge (domestic = 0) \wedge (catsize = 0) Then (type = 3)	1.0	0.6
4	If (aquatic = 1) \wedge (breathes = 0) \wedge (venomous = 0) \wedge (tail = 1) Then (type = 4)	0.7	0.8
5	If (hair = 0) \wedge (airborne = 0) \wedge (aquatic = 1) \wedge (toothed = 1) \wedge (breathes = 1) \wedge (legs = 4) \wedge (catsize = 0) Then (type = 5)	0.9545	0.6667
6	If (airborne = 1) \wedge (fins = 0) \wedge (tail = 0) Then (type = 6)	0.8565	0.8
7	If (hair = 0) \wedge (predator = 1) \wedge (breathes = 0) \wedge (tail = 0) \wedge (domestic = 0) Then (type = 7)	0.8	0.7333

Table 4
Rules generated by NPGA from zoo dataset

Class#	Mined rules	Predictive accuracy	Comprehensibility
1	If (eggs = 0) \wedge (venomous = 0) \wedge (domestic = 0) Then (type = 1)	0.9090	0.8667
2	If (feathers = 1) \wedge (breathes = 1) \wedge (domestic = 0) Then (type = 2)	0.9333	0.8667
3	If (eggs = 1) \wedge (predator = 1) \wedge (toothed = 1) \wedge (catsize = 0) Then (type = 3)	1.0	0.8
4	If (aquatic = 1) \wedge (breathes = 0) \wedge (tail = 1) Then (type = 4)	0.8	0.8667
5	If (airborne = 0) \wedge (aquatic = 1) \wedge (toothed = 1) \wedge (breathes = 1) \wedge (catsize=0) Then (type = 5)	1.0	0.7333
6	If (airborne = 1) \wedge (fins = 0) \wedge (tail = 0) Then (type = 6)	0.8333	0.8
7	If (predator = 1) \wedge (breathes = 0) \wedge (tail = 0) \wedge (domestic = 0) Then (type = 7)	0.875	0.8

Table 5
Rules generated by INPGA from zoo dataset

Class#	Mined rules	Predictive accuracy	Comprehensibility
1	If (eggs = 0) \wedge (venomous = 0) \wedge (domestic = 0) Then (type = 1)	0.9090	0.8667
2	If (feathers = 1) \wedge (breathes = 1) \wedge (domestic = 0) Then (type = 2)	0.9333	0.8667
3	If (eggs = 1) \wedge (predator = 1) \wedge (catsize = 0) Then (type = 3)	0.9879	0.8667
4	If (aquatic = 1) \wedge (breathes = 0) \wedge (tail = 1) Then (type = 4)	0.8	0.8667
5	If (airborne = 0) \wedge (aquatic = 1) \wedge (toothed = 1) \wedge (catsize = 0) Then (type = 5)	1.0	0.8
6	If (airborne = 1) \wedge (fins=0) \wedge (tail = 0) Then (type = 6)	0.8333	0.8
7	If (predator = 1) \wedge (breathes = 0) \wedge (domestic = 0) Then (type = 7)	0.875	0.8667

Table 6
Rules generated by SGA from nursery dataset

Class	Mined rules	Predictive accuracy	Comprehensibility
P	If (parents = usual) \wedge (housing = less_conv) \wedge (social = problematic) \wedge (health=recommended) Then (class = P)	0.7780	0.5
	If (parents = great_pret) \wedge (children = 3) \wedge (social slightly_prob) \wedge (health = recommended) Then (class = P)	0.6892	
NR	If (parents = usual) \wedge (housing = less_conv) \wedge (social = slightly_prob) \wedge (health = not_recom) Then (class = NR)	0.634	0.5
	If (parents = pretentious) \wedge (children = 3) \wedge (housing = convenient) \wedge (health = not_recom) Then (class = NR)	0.7641	
	If (parents = great_pret) \wedge (children = 2) \wedge (housing = critical) \wedge (health = not_recom) Then (class = NR)	0.783	
VR	If (parents = usual) \wedge (housing = less_conv) \wedge (finance = inconv) \wedge (social = slightly_prob) \wedge (health = recommended) Then (class = VR)	0.876	0.378
R	If (has_nurs = proper) \wedge (finance = convenient) \wedge (health = recommended) Then (class = R)	0.81	0.625

Table 7
Rules generated by NPGA from nursery dataset

Class#	Mined rules	Predictive accuracy	Comprehensibility
P	If (parents = usual) \wedge (housing = less_conv) \wedge (social = problematic) Then (class = P)	0.7780	0.625
	If (parents = great_pret) \wedge (social = slightly_prob) \wedge (health = recommended) Then (class = P)	0.8114	
NR	If (parents = usual) \wedge (housing = less_conv) \wedge (social = slightly_prob) \wedge (health = not_recom) Then (class = NR)	0.634	0.5
	If (parents = pretentious) \wedge (children = 3) \wedge (housing = convenient) \wedge (health = not_recom) Then (class = NR)	0.7641	
	If (parents = great_pret) \wedge (children = 2) \wedge (housing = critical) \wedge (health = not_recom) Then (class = NR)	0.783	
VR	If (housing = less_conv) \wedge (finance = inconv) \wedge (social = slightly_prob) \wedge (health recommended) Then (class = VR)	0.897	0.5
R	If (has_nurs = proper) \wedge (finance = convenient) \wedge (health = recommended) Then (class = R)	0.81	0.625

Table 8
Rules generated by INPGA from nursery dataset

Class#	Mined rules	Predictive accuracy	Comprehensibility
P	If (parents = usual) \wedge (housing = less_conv) \wedge (social = problematic) Then (class = P)	0.7780	0.625
	If (parents = great_pret) \wedge (social = slightly_prob) \wedge (health = recommended) Then (class = P)	0.8114	
NR	If (parents = usual) \wedge (housing = less_conv) \wedge (social = slightly_prob) \wedge (health = not_recom) Then (class = NR)	0.634	0.5
	If (parents = pretentious) \wedge (children = 3) \wedge (housing = convenient) \wedge (health = not_recom) Then (class = NR)	0.7641	
	If (parents = great_pret) \wedge (children = 2) \wedge (housing = critical) \wedge (health = not_recom) Then (class = NR)	0.783	
VR	If (housing = less_conv) \wedge (finance = inconv) \wedge (social = slightly_prob) Then (class = VR)	0.897	0.625
R	If (has_nurs = proper) \wedge (finance = convenient) Then (class = R)	0.751	0.75

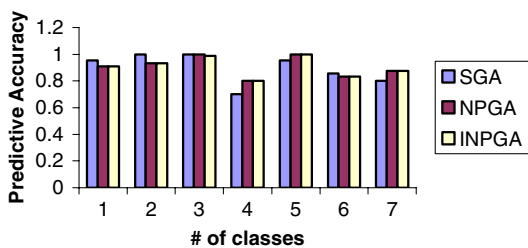


Fig. 5. Predictive accuracy of mined rules by SGA, NPGA and INPGA from zoo dataset.

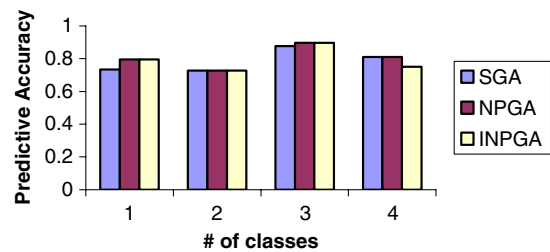


Fig. 7. Predictive accuracy of mined rules by SGA, NPGA and INPGA from nursery dataset.

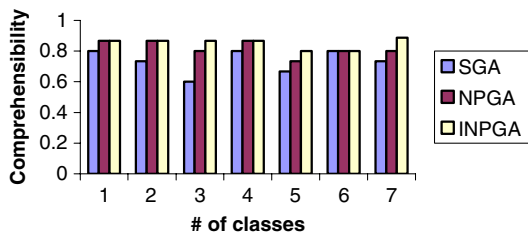


Fig. 6. Comprehensibility of mined rules by SGA, NPGA and INPGA from zoo dataset.

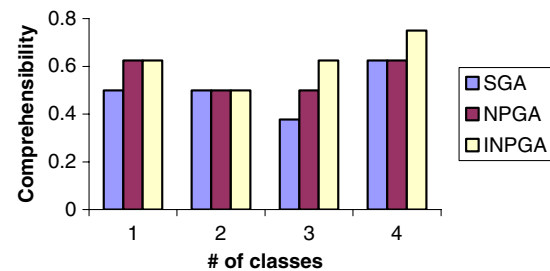


Fig. 8. Comprehensibility of mined rules by SGA, NPGA and INPGA from nursery dataset.

Table 9

Average performance from zoo dataset

Methods	Predictive accuracy	Comprehensibility
SGA	0.89507	0.7333
NPGA	0.9072	0.8191
INPGA	0.9055	0.8476

Table 10

Average performance from nursery dataset

Methods	Predictive accuracy	Comprehensibility
SGA	0.76204	0.50075
NPGA	0.7825	0.5625
INPGA	0.7665	0.625

NPGA and INPGA. Since it is a multi-objective problem we cannot prioritize one objective over to another. Tables 9 and 10 show the average performance of the three methods. If we look at the average performance of the three methods the INPGA performs better than SGA and NPGA.

6. Conclusion

In this paper, we have discussed the use of a multi-objective genetic algorithm for discovering predictive and comprehensible rules [8–10]. We have discussed the basic concepts and principles of application of NPGA and proposed a method called INPGA for classification rule generation and experimental results. Though we have experimented using limited datasets with limited patterns still the results show the trend of performance. The comprehensibility of the discovered rules could, in principle, be improved with a proper modification of the fitness assignment method. We are now concentrating on careful selection of attributes in a preprocessing step [14,15], in order to reduce the number of attributes (and the corresponding search space) given to the INPGA.

Acknowledgements

Authors thank Professor S. Pattanayak of the Institute of Mathematics and Application, Bhubaneswar, and our other colleagues for helpful discussions. Authors also thank the referees for their useful suggestions for improvement of this work.

References

- [1] U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, From data mining to knowledge discovery: an overview, in: U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy (Eds.), *Advances in Knowledge*

Discovery and Data Mining, AAAI/MIT, Cambridge, MA, 1996, pp. 1–34.

- [2] A.A. Freitas, A genetic programming framework for two data mining tasks: classification and generalized rule induction, in: *Genetic Programming 1997: Proceedings of 2nd Annual Conference*, Morgan Kaufman, Los Altos, CA, 1997, pp. 96–101.
- [3] A.A. Freitas, On rule interestingness measures, *Knowledge-Based Systems* 12 (1999) 309–315.
- [4] J.H. Holland, *Adaptation in Natural and Artificial Systems*, Univ. Michigan Press, Ann Arbor, MI, 1975.
- [5] C.M. Fonseca, P.J. Fleming, An overview of evolutionary algorithms in multi-objective optimization, *Evolutionary Computation* 3 (1) (1995) 1–16.
- [6] L. Davis, *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, 1991.
- [7] Z. Michalewicz, *Genetic Algorithms + Data Structure = Evolution Programs*, Springer-Verlag, Berlin, 1994.
- [8] J.D. Schaffer, Some experiments in machine learning using vector evaluated genetic algorithms (doctoral dissertation), Vanderbilt University, Nashville, TN, 1984.
- [9] J.D. Schaffer, Multiple objective optimization with vector evaluated genetic algorithms, in: *Proceedings of the First International Conference on Genetic Algorithms*, 1985, pp. 93–100.
- [10] C.Z. Jainkow, A knowledge intensive genetic algorithm for supervised learning, *Machine Learning* 13 (1993) 189–228.
- [11] A.A. Freitas, A genetic algorithm for generalized rule induction, in: R. Roy et al. (Eds.), *Advances in Soft Computing-Engineering Design and Manufacturing*, Springer-Verlag, New York, 1999, pp. 340–353.
- [12] D. Hand, *Construction and Assessment of Classification Rules*, Wiley, New York, 1997.
- [13] G. Syswerda, Uniform crossover in genetic algorithms, in: *Proceedings of the 3rd International Conference on Genetic Algorithms (ICGA 89)*, 1989, pp. 2–9.
- [14] C.L. Hwang, K. Yoon, *Multiple Attribute Decision Making, Methods and Application, A State of Art Survey*, Springer-Verlag, New York, 1981.
- [15] M. Zeleny, *Multiple Criteria Decision Making*, McGraw-Hill, New York, 1982.
- [16] E. Zitzler, L. Thiele, Multi-objective evolutionary algorithms: a comparative case study and strength Pareto approach, *IEEE Transactions on Evolutionary Computation* 3 (1999) 257–271.
- [17] J. Horn, N. Nafpliotis, E. Goldberg, A niched Pareto genetic algorithm for multi-objective optimization, in: *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, vol. 1, 1994, pp. 82–87.
- [18] M. Laumanns, G. Rudolph, H.P. Schwefel, A spatial predator–prey approach to multi-objective optimization, *Parallel Problem Solving Nature* 5 (1998) 241–249.
- [19] D.E. Goldberg, J. Richardson, Genetic algorithms with sharing for multi-modal function optimization, in: *Proceedings of the 2nd International Conference on Genetic Algorithm*, 1987, pp. 41–49.
- [20] E. Zitzler, L. Thiele, Multi-objective evolutionary algorithms: a comparative case study and strength Pareto approach, *IEEE Transactions on Evolutionary Computation* 3 (1999) 257–271.
- [21] K. Deb, A. Pratap, S. Agrawal, T. Meyarivan, A fast and elitist multi-objective genetic algorithm: NSGA-II, *IEEE Transactions on Evolutionary Computation* 6 (2002) 182–197.
- [22] J.D. Knowles, D.W. Corne, Approximating the non-dominated front using the Pareto archived evolution strategy, *Evolutionary Computation* 8 (2) (2000) 49–172.