Revisiting evolutionary algorithms in feature selection and nonfuzzy/fuzzy rule based classification

Satchidananda Dehuri¹ and Ashish Ghosh²∗

This paper discusses the relevance and possible applications of evolutionary algorithms, particularly genetic algorithms, in the domain of knowledge discovery in databases. Knowledge discovery in databases is a process of discovering knowledge along with its validity, novelty, and potentiality. Various genetic-based feature selection algorithms with their pros and cons are discussed in this article. Rule (a kind of high-level representation of knowledge) discovery from databases, posed as single and multiobjective problems is a difficult optimization problem. Here, we present a review of some of the genetic-based classification rule discovery methods based on fidelity criterion. The intractable nature of fuzzy rule mining using single and multiobjective genetic algorithms reported in the literatures is reviewed. An extensive list of relevant and useful references are given for further research. © 2013 Wiley Periodicals, Inc.

INTRODUCTION

The current information era is characterized by a great expansion in the volume of data that are being generated by low-cost devices (e.g., scanners, bar code readers, sensors) and stored. Intuitively, this large amount of stored data contains valuable hidden knowledge, which could be used to improve the decision-making process of an organization. Piatetsky-Shapiro reported¹ that it is an urgent requirement to develop a semiautomatic tool to discover hidden knowledge. However, discovering knowledge from such a volume of complex data can be characterized as a problem of intractability.² Therefore, the development of efficient and effective tools for revealing valuable knowledge hidden in these databases becomes more critical for enterprise decision making. One of the possible approaches to this problem is by means of data mining or knowledge discovery from databases (KDD).³⁻⁶ Through data mining, interesting knowledge can be extracted and the discovered knowledge can be applied in the target field to increase the working efficiency and to improve the quality of decision making. Some of the knowledge discovery and data mining tools, e.g., DBMiner, DeltaMiner, CN2, which aim at the mainstream of business user are providing up-to-date solutions.⁸,⁹ Interested reader/practitioner can obtain a range of existing state-of-the-art data mining and related tools discussed in Ref 10.

Over last one and half decades, most of the data mining techniques are focused from database perspective. In comparison, little effort has been made from machine learning and soft computing perspective.¹¹ However, recently a growing interest from researchers of evolutionary algorithms and multiobjective evolutionary algorithms for data mining applications are coming up with their own findings. Some of the findings in this direction can be obtained from Refs 12⁻¹⁴. Alcala-Fdez et al.¹⁵ have developed a software tool known as knowledge extraction based on evolutionary learning (KEEL) to assess evolutionary algorithms for the data mining problem of

*Correspondence to: ash@isical.ac.in
1Department of Systems Engineering, Ajou University, Suwon South Korea
2Center for Soft Computing Research, Indian Statistical Institute Kolkata, Kolkata India
DOI: 10.1002/widm.1087
various kinds including regression, classification, unsupervised learning (clustering), and so on. It includes evolutionary learning algorithms based on different approaches: Pittsburgh, Michigan, iterative rule learning (IRL), and genetic cooperative competitive learning (GCCL). Along with the integration of evolutionary learning with different preprocessing techniques, it allowed to perform a complete analysis of any learning model in comparison with existing software tools. Similarly, in recent years, the development of methods for data mining has attracted increasing attention in the fuzzy set community. A systematic discussion of possible benefits of fuzzy methods in data mining is presented in Ref. 25. To this end, this paper presents a well-balanced review of the literature of evolutionary algorithms, hybrid fuzzy genetic rule based system in data mining, and KDD.

Nowadays, the application domain of data mining is getting more complex and complex, i.e., it is shifted from traditional scientific and market basket database mining to biological, health care, agriculture, process monitoring and control, intrusion detection, and social network analysis. For example, detecting unauthorized use, misuse, and attacks that have no previously described patterns on information systems is usually a very complex task for traditional methods. Similarly, data about hospital’s patients might contain interesting knowledge about which kind of patient is more likely to develop a given disease. Hence, viewing all these complexities of the domains and the limitations of statistical classifier, neural network based classifier, decision tree based classifier, and some of the nonparametric classifiers, it is a source of inspiration to develop an intelligent system using various evolutionary algorithms and fuzzy system based approaches. Recall that this paper will review some of the representative data preprocessing using genetic algorithms and classification rule generation using genetic and fuzzy systems.

It should be noted that the quality of the discovered knowledge (whether it is a classification rule, association rule, prediction rule, or clusters) strongly depends on the quality of the data being mined. This has motivated the improvement and development of several data preprocessing techniques such as attribute selection, attribute constructions, and training set selection. The requirements of data preprocessing in KDD and its success are reported in the literature. In the Preliminaries, we present some of the preliminary concepts and definitions of KDD, evolutionary algorithms (EAs), multiobjective evolutionary algorithms (MOEAs) and fuzzy systems. EAs particularly genetic algorithms (GAs) for attribute selection is discussed in the Data Mining Using Genetic Algorithms: Attribute Selection.

Data mining involves various tasks such as classification, clustering, association rule mining, regression, and change detection. Each task can be considered as a problem to be solved by data mining algorithms. In this paper, the utility of GAs for classification task is primarily dealt with. However, the interested reader for other tasks of data mining can refer to Refs 44, 45 for association rule mining based on evolutionary algorithms, for evolutionary algorithms based clustering, for genetic-based regression, and for genetic-based change detection, and so on. Various issues interwined with classification rule mining (CRM) using GAs are discussed in the Data Mining Using Genetic Algorithms.

Later on, fuzzy classification rule mining (FCRM) using genetic and multiobjective genetic algorithm (MOGA) is presented. The aim is to generate a compact set of classification rules by simultaneous optimization of rule accuracy, length of the rules, and number of rules. The last section presents the summary and future research directions.

**PRELIMINARIES**

In this section, some preliminaries are discussed.

**Knowledge Discovery in Databases**

The subject of KDD has evolved and continues to evolve, from the intersection of research from various fields such as databases, machine learning, pattern recognition, statistics, artificial intelligence, reasoning with uncertainties, knowledge acquisition for expert systems, data visualization, machine discovery, high-performance computing, evolutionary computation, multiobjective evolutionary computation, and swarm intelligence. This paper focuses on EAs, particularly GAs, for KDD. Hence, it is important to discuss the definitions and concepts of data mining and the process of KDD.

**Definition:** KDD is defined as the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data.

**KDD Process:** The overall KDD process is interactive and iterative involving four steps: (i) data acquisition and integration, (ii) data preprocessing, (iii) data mining, and (iv) postprocessing. Specifically, the
following steps are interwined in any practical implementation:

**Domain specific knowledge** includes relevant prior knowledge and goals of the application.

**Extracting/selecting the target data set** includes extracting/selecting a data set or focusing on a subset of data instance.

**Data cleansing** includes basic operations, such as noise removal and handling of missing data. Data from real-world sources are often erroneous, incomplete, and inconsistent, perhaps due to operational error or system implementation flaws. Such low-quality data needs to be cleaned prior to mining.

**Data integration** includes integrating multiple, heterogeneous data sources.

**Data reduction and projection** includes finding useful features to represent the data (depending on the goal of the task) and using dimensionality reduction or transformation methods.

**Choosing the function of data mining** includes deciding the purpose of the model derived by data mining algorithms (e.g., summarization, classification, regression, clustering, link analysis, image segmentation/classification, functional dependencies, rule extraction (classification and association rule), or a combination of these).

**Choosing the data mining algorithm(s)** includes selecting method(s) to be used for searching patterns in data, such as deciding on which models and parameters may be appropriate.

**Data mining** includes searching for patterns of interest in a particular representational form or a set of such representations.

**Interpretation** includes interpreting the discovered patterns, as well as the possible visualization of the extracted patterns. One can analyze the patterns automatically or semiautomatically to identify truly interesting/useful patterns for the user.

**Using discovered knowledge** includes incorporating this knowledge into the performance system, taking actions based on knowledge.

**Data Mining:** KDD refers to the overall process of turning low-level data into high-level knowledge. An important step in the KDD process is data mining. Data mining is an interdisciplinary field with a general goal of predicting outcomes and uncovering relationships in data. It uses automated tools employing sophisticated algorithms to discover hidden patterns, associations, anomalies, and/or structures from a large amount of data stored in data warehouses or other information repositories. Data mining tasks can be descriptive, i.e., discovering interesting patterns describing the data, and predictive, i.e., predicting the behavior of the model based on available data. Data mining involves fitting models to or determining patterns from observed data. The fitted models play the role of inferred knowledge. Deciding whether the model reflects useful knowledge or not is a part of the overall KDD process for which subjective human judgment is usually required. Typically, a data mining algorithm constitutes some combination of the following three components.

- **The model:** The function of the model (e.g., classification, clustering, regression) and its representational form (e.g., linear discriminants, neural networks, decision trees). A model contains parameters that are to be determined from the data.

- **The preference criterion:** A basis for preference of one model or a set of parameters over another, depending on the given data. The criterion is usually some form of the goodness-of-fit function of the model to the data, perhaps tempered by a smoothing term to avoid overfitting, or generating a model with too many degrees of freedom to be constrained by the given data.

- **The search algorithm:** The specification of an algorithm for finding particular models and parameters, given the data, model(s), and a preference criterion. A particular data mining algorithm is usually an instantiation of the model/preference/search components.

The more common model functions in current data mining practice include the following: (1) Classification classifies a data item into one of several predefined categorical classes. (2) Regression maps a data item to a real-valued prediction variable. (3) Clustering maps a data item into one of several clusters, where clusters are natural groupings of data items based on similarity metrics or probability density models. (4) Rule generation extracts...
classification rules from the data. (5) Discovering association rules describes association relationship among different attributes. (6) Summarization provides a compact description for a subset of data. (7) Dependency modeling describes significant dependencies among variables. (8) Sequence analysis models sequential patterns, like time-series analysis. The goal is to model the states of the process generating the sequence or to extract and report deviation and trends over time.

The development of new generation algorithms is expected to encompass more diverse sources and types of data that will support mixed-initiative data mining, where human experts collaborate with the computer to form hypotheses and test them.

Evolutionary Algorithms
In essence, the EAs are used to describe computer-based problem-solving algorithms that use computational models of evolutionary processes as key elements in their design and implementation. A variety of evolutionary algorithms have been evolved during the past several years. The major representatives are (i) GAs,65 (ii) evolution strategies (ESs),57,58 (iii) evolutionary programming (EP),59,60 (iv) genetic programming (GP),61,62 (v) estimation of distribution algorithms (EDAs),63 (vi) compact genetic algorithms (CGAs)64 and its variants extended compact genetic algorithms, and (vii) population-based incremental algorithms.65 Each of these constitutes a different approach; however, they are inspired by the same principle of natural evolution. As GAs play a critical role, in this paper it is worth to discuss the basic concepts of genetic algorithms.

A GA is a stochastic method, which have been inspired by Darwin’s theory of evolution. A population of individuals, each representing a possible solution to a problem, is initially created at random. Pairs of individuals combine to produce offspring for the next generation. A mutation operator is used to randomly modify the genetic structure of some individuals of each new generation. The algorithm runs to generate solutions for successive generations. The probability of reproduction of an individual is proportional to the goodness of the solution it represents. Hence the quality of the solutions in successive generations improves. The process is terminated when an acceptable or optimum solution is found or after some predefined time limit. GAs are appropriate for optimization problems, with respect to some computable criterion.

Multiobjective Genetic Algorithms
There are many multiobjective problems requiring simultaneous optimization of several competing objectives. Formally, it can be stated as follows:

We want to find \( \bar{x} = (x_1, x_2, \ldots, x_n) \) in decision space, which maximizes the values of \( p \) objective functions \( F(\bar{x}) = (f_1(\bar{x}), f_2(\bar{x}), f_3(\bar{x}), \ldots, f_p(\bar{x})) \) in objective space within a feasible domain \( \Omega \). Generally, the answer is not a single optimal solution but a set of solutions called a Pareto-optimal set.

Definitions
- A solution vector \( \bar{x} = (x_1, x_2, \ldots, x_n) \) is said to dominate \( \bar{x}' = (x'_1, x'_2, \ldots, x'_n) \) iff \( \bar{x} \) is no worse than \( \bar{x}' \) in all objectives and \( \bar{x} \) is strictly better than \( \bar{x}' \) in at least one objective.
- A solution \( \bar{x} \in \Omega \) is said to be Pareto optimal with respect to \( \Omega \) iff there is no \( \bar{x}' \in \Omega \) for which \( \bar{x}' \) dominates \( \bar{x} \).
- For a given multiobjective problem \( F(\bar{x}) \), the Pareto-optimal set \( P_\epsilon \) is defined as \( P_\epsilon = \{ \bar{x} \in \Omega \mid \exists x'_\in \Omega \text{ for which } \bar{x}' \text{ dominates } \bar{x} \} \).
- For a given multiobjective problem \( F(\bar{x}) \) and Pareto-optimal set \( P_\epsilon \), the Pareto front \( P_f \) is defined as \( P_f = \{ F(\bar{x}) = (f_1(\bar{x}), f_2(\bar{x}), \ldots, f_p(\bar{x})) \mid \bar{x} \in P_\epsilon \} \).

Optimization methods generally try to find a given number of Pareto-optimal solutions, which are uniformly distributed in the Pareto-optimal front. Such solutions provide the decision maker sufficient insight into the problem to make the final decision. Methods such as weighted sum, \( \epsilon \)-constraint, and goal programming have been proposed to search for Pareto-optimal solutions. However, an \textit{a priori} articulation of the preferences to the objectives is required, which is often difficult to decide beforehand. Besides, these methods can only find one solution at a time. Other solutions cannot be obtained without recomputation with the free parameters reset.

In contrast, GAs maintain a population and thus can search for many nondominated 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 number of multiobjective GAs (MOGAs)\(^{44,66} \) have been introduced in the literature. 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 fitness. Schaffer’s VEGA\(^{67}\) is such a method. The Predator–prey approach\(^{68}\) 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 nondominated individuals in the population are regarded as fittest regardless of their single objective values. Since Pareto-based approaches respect better the dominant nature of multiobjective problems, their performance is reported to be better.

A diversity maintenance strategy works by distributing the solutions uniformly in the Pareto front, instead of accumulating solutions in a small region only. Fitness sharing\(^{66}\) is another way to maintain diversity in the Pareto front by sharing the fitness of an individual with its neighborhood. Restricted mating is another alternative, where mating is permitted only when the distance between two parents are large enough. Pareto archive evolution strategies (PAES),\(^{69}\) strength Pareto evolutionary algorithms (SPEA),\(^{68}\) and nondominated sorting genetic algorithms-II (NSGA-II)\(^{66}\) are some of the front-end MOGAs. They all adopt Pareto-based fitness assignment strategy and implement elitism. A good comprehensive study of MOGA can also be found in Ref 13.

Let us discuss the working principles of NSGA-II and SPEA here as they are the most popular in the literature.

**Nondominated Sorting Genetic Algorithm-II:** The NSGA-II procedure\(^{70}\) attempts to find multiple Pareto-optimal solutions of a multiobjective optimization problem by using an elitist mechanism, explicit diversity preserving mechanism, and by emphasizing on nondominated solutions. At any generation \(t\), the offspring population is first created by using the parent population (size \(N\)) and the usual genetic operators. Thereafter, the two populations are combined together to form a new population of size \(2N\). Then, the new population is classified into different nondominated fronts. Thereafter, the next population is filled by points of different nondominated fronts, one at a time. Since the overall population size of new population is \(2N\), not all fronts can be accommodated in \(N\) slots available for the next population. All fronts that could not be accommodated are deleted. When the last allowed front is being considered, there may exist more points in the front than the remaining slots in the new population. Instead of arbitrarily discarding some members from the last front, the points that make the diversity of the selected points are chosen.

**Strength Pareto Evolutionary Algorithms:** SPEA\(^{71}\) maintains an external population at every generation storing all nondominated solutions obtained so far. At each generation, external population is mixed with the current population. All nondominated solutions in the mixed population are assigned fitness based on the number of solutions they dominate. Dominated solutions are assigned fitness worse than the worst fitness of any nondominated solutions. A deterministic clustering technique is used to ensure diversity among nondominated solutions. One of its variants is SPEA2\(^{72}\).

**Fuzzy Set Theory**

**Fuzzy Sets:** Unlike a classical set whose boundary is clearly defined, fuzzy set boundaries are not clearly defined. An object belongs to a fuzzy set to a certain degree, called the degree of membership, typically represented by a real-valued number in the interval \([0, 1]\) (i.e., \(\mu_A: A \rightarrow [0, 1]\), where \(\mu_A\) is the membership function (MF)). In contrast, the MF \(\mu_A\) for a given set is a map of the form \(\mu_A: A \rightarrow \{0, 1\}\). Without loss of generality, a fuzzy set is a generalization of a classical set. In case of the fuzzy set, the cardinality of a set \(A\) can be defined as

\[
|A| = \sum_{x \in X} \mu_A(x). \tag{1}
\]

Operations on fuzzy sets are generalizations of operations on sets. This generalization can be done in several different ways, and details can be found in Ref 73.

**Membership Functions:** In fuzzy classifiers, the range of a continuous feature is divided into several intervals. Then each interval is considered to be a fuzzy set and an associated MF is defined. Thus, the input space is divided into several subregions that are parallel to input axes. For each subregion, a fuzzy rule is defined; if the input is in the subregion, then it belongs to the class associated with that subregion. For an unknown input pattern, the degree of membership corresponding to all fuzzy sets is calculated and the input is classified into the class with maximum degree of membership. Hence, the MFs directly influence the performance of the fuzzy classifier.

MFs can be defined in a number of ways, based on different shapes and different number of parameters. Commonly used MFs\(^{74}\) are triangular, trapezoidal, Gaussian, sigmoidal, and reverse-sigmoidal. Some of the multidimensional MFs are rectangular.
DATA MINING USING GENETIC ALGORITHMS: ATTRIBUTE SELECTION

Feature selection is one of the data-preprocessing tasks, which brought attention to the researchers. It is the process of selecting a subset of available features to use for constructing the model of interest. Solution to the feature selection problem is neither trivial nor unique. The set of optimal features can be different for different hypothesis spaces. Therefore, optimality of a feature subset should only be defined in the context of the family of admissible modeling functions from which it is intended to select the one that is finally deployed. As the dimensionality of a domain expands, the size of the search space grows exponentially. The number of candidate feature subsets is \(2^N\), where \(N\) is the number of features. Therefore, finding an optimal feature subset is usually intractable\(^2\) and many problems related to feature selection have been shown to be NP-hard.\(^2\)

It can be defined as “the feature selection problem involves the selection of a subset of \(d\) features from a total of \(N\) features, based on a given optimization criterion (here it is denoted as \(J\)).” Let us denote the \(N\) features uniquely by distinct numbers from 1 to \(N\), so that the total set of \(N\) features can be written as \(U = \{1, 2, 3, \ldots, N\}\). \(X\) denotes the subset of selected features, and \(Y\) denotes the set of remaining features. Therefore, \(U = X \cup Y\) at any time. \(J(X)\) denotes a function evaluating the performance of \(X\). \(J\) may evaluate either the accuracy of a specific classifier on a specific data set (e.g., the wrapper approach as in Ref 76) or a generic statistical measurement (e.g., the filter approach\(^77\)). The choice of evaluation function, \(J\), depends on the particular application.

Interest in feature selection is increased due to several reasons. Reasons such as new applications dealing with vast amount of data have been developed, e.g., data mining,\(^78,79\) multimedia information retrieval,\(^80,81\) and medical data processing.\(^82\) Since the first processing of a large volume of data is critical in these applications for the purpose of real-time processing or to provide a quick response to users, limiting the number of features is a very important requirement. Feature selection is a prerequisite when using multiple sets of features, as this is required for the subsequent processing involving classification or clustering. Some examples include aerial photo interpretation,\(^83\) correspondence in stereo vision,\(^83\) and handwriting recognition.\(^84\)

Feature selection can be broadly categorized into two types: (i) filter approach and (ii) wrapper approach. In this work, we will give more emphasis on the wrapper approach rather than the filter approach. More details about feature selection can be found in Ref 85.

Filter Approach

In this approach, feature selection is performed without taking into account the classification algorithm to measure how good is the candidate feature subset. Here, the main goal is to select a subset of features that preserves as much as possible the relevant information found in the entire set of features. One example of a filter method for feature selection can be found in Ref 77. The basic idea of this method is to use a vertical compactness criterion for evaluating the quality of a given candidate feature subset. The method starts by assuming that one has some idea about how much inconsistency can be tolerated in the data being mined. The term inconsistency refers to the situation where two or more data instances have the same value for all selected features but have different goal attribute values (classes). Once a maximum inconsistency rate is specified, the method searches for a feature subset that produces the minimum number of projected instances, provided that the resulting inconsistency rate is not greater than the specified maximum. The projected instances of a given feature subset \(X\) are the instances produced by eliminating all features not specified in \(X\) and then eliminating duplicate instances from what is left. The challenge is to find an attribute subset that produces the least possible number of projected instances, provided that the resulting inconsistency rate is not greater than a specified maximum. Some of the feature selection methods based on the filter approach can be found in Refs 86, 87. Sanchez-Marono et al.\(^87\) have contributed a good comparative study on filter methods for feature selection.

Wrapper Approach

In this approach, feature selection is performed by taking into account the classification algorithm that will be applied to the selected features. Here, the goal is to select a subset of features that is optimized for a given classification algorithm. The following procedure gives a clear indication of how the wrapper approach proceeds.

pyramidal, truncated rectangular pyramidal, polyhedral pyramidal, and bell shaped.
Procedure_Wrapper()

1. Input all features.
2. Generate a candidate feature subset.
3. Create a model with the feature subset generated in Step 2.
4. Measure the performance of the model.
5. If performance is found satisfactory, then go to Step 6; otherwise go to Step 2.
6. Output a feature subset.

Note that in the wrapper-based approach, the classification algorithm is run many times, each time with a different subset of features. The performance of the classification algorithm in each run is used to evaluate the quality of the corresponding feature set. One important point about this approach is that the performance of the classification algorithm—which is used to evaluate the quality of the feature subset—cannot be evaluated on the test set. To avoid this, one must reserve a part of the training set for the purpose of evaluating the performance of the classification algorithm within the loop of the feature selection procedure. One simple way of doing this consists of randomly dividing the training set into two subsets: one for training and the other for testing. The former is used to train the classification algorithm. Once the algorithm is trained, its performance will be measured on the test set. This data set plays the role of an unseen test set for the classification algorithm. Indeed, when the feature selection method terminates, the best feature subset found by that method is given to the classification algorithm, which is then finally run on the entire training set. The knowledge discovered by the classification algorithm is then evaluated on the test set, whose data instances remained unseen during the entire run of the feature selection method.

So far our discussion was focused on how to use a classification algorithm for evaluating the quality of a candidate feature subset. We now turn to the problem of how to generate candidate feature subsets to be evaluated. Clearly, if the data being mined have a small number of attributes, we can generate all possible subsets and measure the performance of the classification algorithm on each of these subsets. Unfortunately, the number of candidate feature subsets grows exponentially with the number of available features, as mentioned above. But in data mining, the number of features is large enough, so it is not practicable to apply the exhaustive search procedure to generate and evaluate every possible feature subset.

Genetic Algorithms for Feature Selection

Among the various categories of feature selection algorithms, the evolutionary algorithms particularly GAs are popular and widely used. Furthermore, GA is naturally applicable to feature selection over nonevolutionary-based approaches because the problem has an exponential search space. The individual encoding and fitness functions are two important steps to be determined before discussing about the details of GA for feature selection. The other genetic operators are the same as the standard GA.

Individual Encoding

To the best of our knowledge, in GA-based feature selection two individual encoding mechanisms are used.

Binary Individual Encoding: The search space of a feature selection problem consists of all possible feature subsets. Each state in this search space can be represented by a fixed length string containing $N$ bits, where $N$ is the number of available features. The $i$th bit, $i = 1, 2, \ldots, N$, indicates whether or not feature $F_i$ is selected. As an example, the chromosome of length 8 represented by 00101000 means that the third and fifth features are selected. That is, the chromosome represents only third and fifth features and rest of the features are not selected for this particular chromosome.

The main advantage of this encoding scheme is its simplicity. Actually, some authors have emphasized that when using this approach there is no need to develop problem-dependent genetic operators. Any standard crossover and mutation operators developed for fixed-length binary strings will do. However, it does not imply that they are the best choice.

Index-Based Individual Encoding: An alternative form of individual encoding for feature selection was proposed in Ref 88. In this case, an individual consists of $m$ genes, where each gene can contain either the index (id) of an attribute or a flag-say 0—indicating no attribute. The value of $m$ is a user-specified parameter. An attribute is considered selected if its corresponding index occurs in at least one of the genes of the individual. For instance, consider the following individual, where $m = 5$: $0F_1F_40F_1$, in this case only features $F_1$ and $F_4$ are selected. The fact that $F_1$ occurs twice in the individual, whereas $F_4$ occurs only once, is irrelevant for the purpose of decoding the individual into a subset of selected features.

Two motivations for this unconventional individual encoding are as follows. First, the fact that an attribute can occur more than once in an individual that can act as a redundancy mechanism that
increases robustness and slows the loss of genetic diversity. Second, in this encoding, the length of an individual is independent of original attributes. Hence, this approach is more scalable to data sets with a very large number of attributes. In this case, we can specify a number of genes much smaller than the number of original attributes, which would not be possible with the standard binary encoding for attribute selection.

The use of such an individual encoding suggests the development of new genetic operator tailored for this kind of encoding. Hence, in addition to crossover and mutation operators, the GA uses a new kind of genetic operator called delete_feature(). This operator accepts as input one parent and produces as output one offspring where all occurrences of duplicate attributes are removed from the parent. Therefore, this operator has a bias that favors the selection of smaller attribute subsets. For instance, in the above example of individual, if feature $F_1$ is chosen to be deleted then the individual will look like 00000.

**Fitness Function**

As stated above, in the wrapper-based approach, the fitness function of a GA for feature selection involves a measure of performance of a classification algorithm using only the subset of features selected by the corresponding GA individual. This basic idea is illustrated in the following procedure:

**Genetic_Wrapper_Feature()**

1. Create a population of individuals with a selected set of features.
2. Run classification algorithm using these individuals.
3. Compute the fitness of each individual based on performance measure.
4. Select the fittest individuals based on any of the standard selection mechanisms.
5. Apply genetic operators like crossover and mutation.
6. Test the performance of the individuals, if satisfactory then stop and exit, otherwise go to Step 2.

According to this simplicity, the vast majority of GAs for feature selection follows the wrapper approach. There is, however, an alternative view, in which a GA often has aspects of both wrapper and filter approaches, depending on what is measured in the fitness function. If the fitness function involves only a measure of performance of the classifier, e.g., the classification accuracy rate, then the GA is definitely following a wrapper-based approach.

Now suppose the fitness function involves both the classifier accuracy rate and the number of selected features. The value of these criteria is independent of the classifier. It depends only on the genes of an individual. Hence, the use of this criterion adds to the fitness function a certain flavor of the filter approach, even though one might argue that the wrapper criterion of classifiers’ accuracy rate is still predominating in most cases. The predominance of the wrapper criterion is normally due to the use of a fitness function, which is a weighted sum of classification accuracy and the number of selected attributes, where the weight of the former term is usually greater than the weight of the latter. An example of this kind of the fitness function for attribute selection is proposed in Ref. 89. This function takes the form:

$$\text{fitness}(s) = \text{info}(s) - \text{cardinality}(s)$$

$$+ \text{accuracy}(s),$$

(2)

where $s$ is the candidate feature subset associated with an individual, $\text{accuracy}(s)$ is the classification accuracy rate of a classification algorithm using only the attributes in $s$, $\text{cardinality}(s)$ is the number of attributes $s$, and $\text{info}(s)$ is an information theoretic measure, which is an estimate of the discriminatory power of the features in $s$. Note that $\text{cardinality}(s)$ and $\text{info}(s)$ take on values independent of the classification algorithm used, so that they are filter-oriented criteria.

It is also possible to define a fitness function that follows a pure filter approach, ignoring altogether the classification accuracy rate of the classifier. This approach normally has the advantage of reducing the processing time of the GA. This stems from the fact that in general the computation of a filter-oriented criterion for an attribute subset is considerably faster than running a classification algorithm with that attribute subset, as required in the wrapper approach.

One example of a GA following the filter approach can be found in the above-mentioned work.\(^{89}\) In one of the experiments reported by the authors, the wrapper component of the fitness function, $\text{accuracy}(s)$, was switched off, so that the GA effectively followed a purely filter approach. Unfortunately, this purely filter variant did not produce good results.

A summary of the main aspects of fitness functions for feature selection involving a wrapper criterion is as follows:

Bala et al., in 1995,\(^{90}\) used decision tree as a classification algorithm. Predictive accuracy and the number of selected features are considered as the fitness function. In 1996, they have extended it by introducing one more criteria in fitness function known.
as info. Chen et al.\(^9\) in 1999 used predictive accuracy and the number of selected features in the fitness function by considering Euclidean decision table as the classification algorithm. Guerra-Salcedo and Whitely in 1998\(^9\) used predictive accuracy as the fitness function and Euclidean decision table as the classification algorithm. Cherkauer and Shavlik in 1996\(^8\) used decision tree as the classification algorithm, and predictive accuracy, the number of selected features, and average decision tree-size are three criteria for the fitness function. Yang and Honavar, in 1998,\(^9\) used neural networks for classification, in which the predictive accuracy and attribute cost are the two objectives used in the fitness function. Moser and Murty, in 2000,\(^4\) used the predictive accuracy and number of selected attributes as the fitness function and nearest neighbor techniques as the classification algorithm. Ishibuchi and Nakashima, in 2000,\(^5\) used predictive accuracy, the number of selected attributes, and the number of selected instances as the criteria for fitness, and nearest neighbor as the classification algorithm.

It should be noted that so far our discussion was focused on GAs for selecting attributes for a single classification algorithm, or classifier. However, a GA can also be used for selecting features by considering many classifiers (e.g., ensembling classifiers) to maximize predictive accuracy. A GA that performs feature selection for generating an ensemble of classifiers is discussed in Ref 96. The basic idea is that a GA for feature selection is run many times, and each run selects a subset of features. Then each of the selected feature subsets is given for a classifier of the ensemble, so that different classifiers of the ensemble are trained with different feature subsets. Some of the other potential contributions in this directions are presented in Refs 97–100.

### DATA MINING USING GENETIC ALGORITHMS

Recall that data mining is one of the important steps of the KDD process and since in the present article we are interested in CRM therefore the discussion of other data mining tasks is not considered. However, the interested reader can refer to Ref 3 for more about the data mining tasks.

#### Classification Rule Mining

This task has been studied for many decades by the machine learning and statistics communities.\(^1\) In this task, the goal is to predict the value (the class) of a user-specified goal attribute based on the values of other attributes, called predicting attributes. Classification rules can be considered as a particular kind of prediction rule where the rule antecedent (“IF” part) contains predicting attribute and rule consequent (“THEN” part) contains a predicted value for the goal attribute. Alternatively, the CRM can be considered as a task to uncover knowledge, which is represented in the form of IF–THEN statement.

**RULE:** IF \(cond_1\) AND \(cond_2\) AND \(\ldots\) AND \(cond_m\) THEN Class_value, where each of the conditions in the rule antecedent part can be written as \(attribute_i \text{ value}_{ij}\), \(attribute_i\) denotes the \(i\)th attribute in the set of predictor attributes, \(value_{ij}\) denotes the \(j\)th value of the domain of attribute \(i\), and OP is a comparison operator—usually in \(\{\ =\ ,\ =
\neq\ ,\ >\ ,\ <\ ,\ \leq\ ,\ \geq\ \}\).

The classification rule can also be derived from a decision tree.\(^3\) The decision tree induction algorithm is one of the most successful learning algorithms, due to its various attractive features: simplicity, comprehensibility, and the absence of parameters.\(^10\) The various improvements such as ID3,\(^10\) ID4,\(^10\) ID5,\(^10\) \(\text{ITI}\),\(^10\) \(\text{C4.5}\),\(^10\) and \(\text{CART}\)\(^10\) over the original decision tree algorithms have been proposed.

**Example:** The table given below shows an example of a very small training set with 14 samples, four predictor attributes, namely age, income, student, and credit_rating, and one class attribute, namely buys_computer. Based on this training instances, the objective of a CRM algorithm is to discover rules that predict the value of buys_computer for test data instances.

<table>
<thead>
<tr>
<th>Age</th>
<th>Income</th>
<th>Student</th>
<th>Credit_rating</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>≤30</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>≤30</td>
<td>high</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>31…40</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>31…40</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>≤30</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>≤30</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>≤30</td>
<td>medium</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>31…40</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>31…40</td>
<td>high</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
</tbody>
</table>

Based on the basic decision tree algorithm- after recursive partitioning of each of the subsets (i.e., until all samples for a given node belong to the same class,
or there are no remaining attributes on which the samples may be further partitioned, or there are no samples for the branch of a test attribute) the entire tree can be generated and is illustrated in Figure 1.

The IF–THEN classification rules can be derived by tracing the path from the root node to each leaf node in the tree. The rules extracted from Figure 1 are as follows:

IF age = "≤30" AND student = "no" THEN class = "no"
IF age = "≤30" AND student = "yes" THEN class = "yes"
IF age = "31...40" THEN class = "yes"
IF age = ">40" AND credit_rating = "excellent" THEN class = "no"
IF age = ">40" AND credit_rating = "fair" THEN class = "yes"

Although the generated rules are comprehensible with an acceptable accuracy, the decision tree algorithms do not handle two types of exceptions, namely (i) when the information gain for two or more attributes is the same; (ii) when two or more classes have equal probabilities in a tree leaf. In addition, if the attribute is not discretized then it must be discretized with optimal intervals; otherwise, the branching factor will be more. Furthermore, if the dimension of the data set is more, which is common in data mining, particularly biological data, then the depth of the tree will be more.

Another problem associated with conventional decision tree building algorithms is that they perform a greedy search for attributes to be put into the tree. By greedy search, we mean this kind of algorithms that builds a tree in a step-by-step fashion, adding one attribute at a time to the current partial tree, and at each step the best possible local choice is made. Note, however, that a sequence of best possible local decisions does not guarantee the best possible global decision. This limitation is one of the motivations to use genetic algorithms to discover classification rules.44,110

**Genetic Algorithms for CRM**

In this subsection, we discuss several issues along with problem-specific operators, fitness functions, individual representation, and population initializations/seeding related to developing GAs for CRM. In addition, some of the recent publications are cited for advancement of this area.

**Genetic Representations**

Earlier work of GAs addresses that each individual corresponds to a candidate solution to a given problem. However, as the problem is to mine an optimal set of classification rules rather than a single rule, it is very important to discuss how to encode a set of rules in a GA population. There are two approaches to accomplish this task: Michigan and Pittsburg.

**Pittsburg versus Michigan:** In Pittsburg approach, each individual of the GA population represents a set of classification rules, i.e. an entire candidate solution. In contrast, in the Michigan approach each individual represents a single classification rule.

In the Michigan approach, there are at least two possibilities for discovering a set of rules. First one says, let each run of the GA discover a single rule (the best individual produced in all generations) and simply run the GA multiple times to discover a set of rules. An obvious disadvantage of this strategy is that it is computationally expensive, requiring many GA runs. The second possibility is to design a more elaborate GA where a set of individuals, possibly the whole population, corresponds to a set of rules.

In the Pittsburg approach, since an individual represents a rule set, we can solve the rule interaction by using a fitness function that directly measures the performance of the rule set as a whole. On the other hand, in this approach an individual tends to be significantly more complex, at least syntactically longer, than in the Michigan approach. This often leads to more complex genetic operators. On the other hand, in the Michigan approach, since an individual represents a single rule, it is syntactically shorter and the genetic operators can be simpler. However, the problem of rule interaction is ignored. Hence, assuming that we want to discover a set of rules, in the
Michigan approach we often need to add to the GA some method(s) that foster the discovery of a good set of rules, rather than converging to a single good rule. One way of preventing this undesirable convergence is to use some kind of niching mechanism.\textsuperscript{111,112} For instance, fitness sharing was successfully used in a GA for predicting rare events.\textsuperscript{113}

A further discussion about the Michigan and Pittsburg approaches can be found in Refs 22, 114, 115. It is also possible to use the hybrid Michigan/Pittsburg approach for rule mining. In particular, Ref 116 proposes a GA where the setting of some parameters can be varied to explore different combinations of the Michigan and Pittsburg approaches.

Throughout this paper, we will refer to the encoding of a single rule into an individual so that we are implicitly assuming the use of Michigan approach.

Each individual in the population represents a candidate rule of the form “if antecedent then consequent.” 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 \( A_i = V_j \), where \( A_i \) is the \( i \)th attribute and \( V_j \) is the \( j \)th value of the \( i \)th attribute’s domain. The consequent consists of a single condition of the form \( G = g_l \), where \( G \) is the goal attribute and \( g_l \) is the \( l \)th value of the goal attribute’s domain.

When using a binary (low-level) encoding, in general an attribute will be assigned a certain number of bits, which depends on the data type of the attribute (i.e., whether categorical or continuous). Some of the GAs for rule discovery that use binary encoding can be found in Refs 116–118 along with their pros and cons. High-level encoding seems particularly advantageous in the case of continuous attributes, where a binary encoding tends to be somewhat cumbersome and/or inefficient, particularly for a large number of continuous attributes, as discussed above. For more details, readers can refer to Ref 13.

Note that a hybrid low- and high-level encoding can also be used. Indeed such a hybrid encoding seems suitable for representing classifications rules involving mixed data. One such reference can be obtained from Ref 119.

A string of fixed size encodes an individual with \( n \) genes representing the values that each attribute can assume in the rule. In addition, each gene also contains a Boolean flag \( f_p \) except the \( n \)th gene that indicates whether or not the \( n \)th condition is present in the rule antecedent. Hence although all individuals have the same genome length, different individuals represent rules of different lengths.

Note that this approach implicitly assumes a positional encoding of attributes in the genotype. Hence this simplifies the action of genetic operators such as crossover. By choosing the same crossover points in both parents, we can directly swap the corresponding genetic material between them without producing some kinds of invalid offspring. In contrast, the use of variable-length genotype directly may have problem at the time of crossover. Details with an extensive examples can be obtained in Refs 120, 121.

Similarly, the goal attribute is also encoded in the individual. This is one possibility. Choosing the best rule consequent on the fly is another possibility.\textsuperscript{22} Note that both the approaches produce offspring with lower fitness than the parents after a crossover operator. This is due to the fact that different individuals of the population can be associated with different rule consequences. This is a point that deserves more research efforts to apply successfully. One of the solutions is to use some kinds of speciation mechanism,\textsuperscript{122} where only individuals with the same specie (in this case having the same rule consequent) can mate with each other.

The third possibility is to associate all individuals of the population with the same predicted class, which is never modified during the execution of the algorithm. Hence if we want to discover a set of classification rules predicting \( k \) different classes, we would need to run the evolutionary algorithm at least \( k \) times, so that in the \( i \)th run, \( i = 1, 2, 3 \ldots, k \), the algorithm discovers only rules predicting the \( i \)th class.

**Fitness Function**

Since the general goal of data mining is to extract knowledge from data, it is important to bear in mind some desirable properties of discovered knowledge. Namely discovered knowledge should be accurate, comprehensible, and interesting. Let us discuss how these criteria can be defined and used in the fitness evaluation of individuals in GAs.

**Comprehensibility Metric**: There are various ways to quantitatively measure rule comprehensibility. A 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 comprehensibility decreases. When using a GA for classification rule discovery, the measure of rule comprehensibility can be easily incorporated into a fitness function. In general, this is done by using a weighted fitness function, with a term measuring predictive accuracy and another term measuring rule comprehensibility, where each term has a user-defined weight. Variants of this basic idea are also used.\textsuperscript{115,123} Although the objective approach is easy to implement,
but it is ignoring the subjective aspects of rule comprehensibility. One approach that does take this subjective aspect into account consists of using a kind of interactive fitness function.\textsuperscript{124}

**Predictive Accuracy:** As already mentioned, our rules are of the form IF $A$ 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 is

$$PA = \frac{|A \& C|}{|A|},$$

where $|A \& C|$ is defined as the number of records satisfying both $A$ and $C$.

Alternatively, the performance of a classification rule with respect to predictive accuracy can be summarized by a confusion matrix.\textsuperscript{3,125} In addition, sometimes it is desirable to use application-dependent fitness functions that are not only related to predictive accuracy but also implement a more direct measure of profitability for the user (e.g., Ref 126).

Finally, let us discuss the difficult issue of rule interestingness.

**Interestingness:** Rule interestingness is measured by two approaches: (i) subjective and (ii) objective. The best judge of subjective approach is the human user. By the interactive fitness function, one can implement the subjective measure of the interestingness of a rule/rule set. Romao et al.\textsuperscript{127} proposed a kind of subjective approach, where the fitness function evaluates both the predictive accuracy and the surprisingness of the rule represented by an individual.

Alternatively, one can use an objective approach, where the fitness function incorporates an objective measure of rule interestingness. Some of the proposed methods in this direction can be obtained in Ref 128.

The overall fitness is computed as the arithmetic weighted mean

$$f(\text{rule}) = \frac{w_1 \cdot \text{rule}_{\text{comp}} + w_2 \cdot PA + w_3 \cdot \text{rule}_{\text{inter}}}{w_1 + w_2 + w_3},$$

where $w_1$, $w_2$, and $w_3$ are user-defined weights. $\text{rule}_{\text{comp}}$ is the measure of the comprehensibility, $PA$ is the predictive accuracy, and $\text{rule}_{\text{inter}}$ is the interestingness of the rule, respectively.

Although the weighted fitness function is popular in the literature, it is not always a good approach. In particular, when there are several criteria to be evaluated, it is often the case that these criteria are conflicting and or noncommensurable. For instance, predictive accuracy and rule comprehensibility are often conflicting criteria (improving a rule with respect to one of them can worsen the rule with respect to the other). The three identified criteria such as predictive accuracy, comprehensibility, and rule interestingness are intuitively noncommensurable and can be conflicting in many cases. This suggests the use of multiobjective approach for rule discovery. Some of the proposals in this direction with promising solutions can be obtained in Refs 120, 121. A good comprehensive review of multiobjective evolutionary algorithms for rule discovery can found in Ref 13.

**Genetic Operators**

One can also consider the idea of uniform crossover.\textsuperscript{129} However, depending on whether a rule is too specific or too general, one can design/adopt special kind of crossover to generalize or specialize a given rule. After crossover is complete, the algorithm analyzes if any invalid individual is created. If so, a repair operator is used to produce valid 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 influence the comprehensibility of the rules. These operators randomly insert and remove, a condition in the rule antecedent. These operators are not part of the regular GA.

**DATA MINING USING FUZZY-GENETIC APPROACH**

Classification tasks of data mining using the fuzzy approach is becoming very popular day by day as it uses linguistic variables, which are very close to human descriptions of structure in data.\textsuperscript{130} In addition, fuzzy rule based systems are highly capable of handling a nonlinear problem of classification. So how the classification problem is handled is an important research topic in fuzzy classification systems.\textsuperscript{131–134}

Fuzzy systems themselves do not exhibit learning capabilities. EAs particularly GAs can be used as a learning algorithm of the fuzzy systems.\textsuperscript{135} In other words, how to adapt GAs for mining FCRs is the focus of this section.

**Fuzzy Classification Rule Mining**

In contrast to classification rules, FCRs are popular because of their comprehensibility,\textsuperscript{136,137} (i.e., each fuzzy if–then rule is interpreted via linguistic values such as large, medium, and so on) and higher
classification ability. Fuzzy rules for classification systems are obtained via two available approaches: (i) directly by experts and (ii) through an automatic learning process. Several methods have been introduced for fuzzy classification.138–144

Without loss of generality, in FCRM, the ranges of input variables are divided into subregions in advance and for each grid region a fuzzy rule is defined. We call the subregions fuzzy regions. Then a fuzzy set with a MF is defined for each subinterval. The MF defines the degree that the input belongs to the fuzzy set.

In general, the fuzzy rule for classification problems in a d-dimensional space is represented as follows:

\[
\text{IF } x_1 \text{ is } A_1 \text{ and } x_2 \text{ is } A_2 \text{ and } \ldots \text{ and } x_d \text{ is } A_d \text{ THEN class } C,
\]

where \(<x_1, x_2, \ldots, x_d>\) is a d-dimensional pattern vector, \(A_i\), \(i = 1(1)d\) is an antecedent fuzzy set and \(C\) is the class label of the rule.

The if part is connected by the AND operator, and each fuzzy rule is connected by the OR operator. These fuzzy rules are defined according to the experts’ knowledge.

The following pseudocode is used for classification:

**Algorithm for CRM()**

### Classifier Design

- Divide the input space into subregions.
- Assign a fuzzy set with a MF to each subregion.
- Define the fuzzy rules according to the experts’ knowledge.

### Classification

- For each fuzzy rule, considering the if part, compute the degree of membership of each input variable and perform the AND operation to find out the minimum value.
- Considering the OR operation among fuzzy rules, the rule having degree of membership is larger then assign the unknown sample to the class associated with that rule.

Although it is understandable, it has lots of drawbacks. Usually it is difficult to acquire knowledge from experts. If acquired, performance of the resulting classifier is far from satisfaction. At the same time, it is very difficult to divide the input space into regions beforehand. We do not know to what extent we need to divide the input space, and the size of the division should be determined not for the range of the input variable but for each class that is approximated. In addition, if some of the input variables are correlated it is inadequate to approximate the region by a rectangle, which is parallel to the input axes.

To solve the problems, several fuzzy classifiers with learning capability have been developed. These fuzzy classifiers extract fuzzy rules with variable size fuzzy regions from data, and fuzzy regions are not necessarily rectangles parallel to the input axes.

### FCRM with Learning Capability

In FCRM, how many fuzzy rules must be generated to realize sufficient recognition rates for both the training data and test data sets are really posed as a challenging problem. The reason is that it depends on how data for each class are distributed and how data between different classes overlap in the input space.

For example, if class \(i\) is approximated by one fuzzy region, then one fuzzy rule is sufficient for class \(i\). But if this does not hold, we need to define more than one fuzzy rule to resolve overlaps between classes and then to improve the recognition rates. There are several ways to solve this problem by introducing learning. Two of them are as follows: (i) detect overlaps, if they exist then generate an additional fuzzy rules or modify existing fuzzy rules to resolve the overlaps; (ii) generate fuzzy rules without considering overlaps and then to tune MFs for overlap resolution. In the latter method, class data are divided into clusters in advance (preclustering) or after rule generation (post-clustering).

The following steps are required for the rule generation:

1. Generate fuzzy rule using all or part of the training data included in a class. If there are no data remained to generate rules, go to Step 3.
2. Check whether the fuzzy region defined in Step 1 overlaps with other fuzzy regions defined previously. If there is no overlap, go to Step 1. Otherwise resolve the overlap by modifying fuzzy rules and go to Step 1.
3. Tune MFs so that the recognition rate of the training data is improved.

In the preclustering method, first the training data belonging to a class are divided into several clusters and then for each cluster a fuzzy rule is generated. While in the postclustering method, first one fuzzy rule is generated for each class and then if the
FIGURE 2 Approaches for FCRM using GAs.

The recognition rate is not sufficient fuzzy rules are generated to resolve overlaps between classes. After rule generation by preclustering and postclustering, we can improve the recognition rate by tuning fuzzy rules, i.e., locations and slopes of the MFs using the training data.

Many approaches have been proposed for FCR generation, such as heuristic procedures, neural fuzzy techniques, clustering methods, and GAs. In this paper, we are restricting ourselves with GA-based FCRM.

GAs for FCRM

In past few decades, GAs have been employed for fuzzy if–then rules and adjusting MFs of fuzzy sets. In this section, we will focus on fuzzy if–then rule generation (specifically for classification) using GAs. Figure 2 shows a taxonomy of GAs for FCR mining.

From the taxonomy, one can conclude that there are at least three different approaches to generate FCRs. The first approach is a priori fixing the MFs and generating FCRs by GAs, whereas in the second approach we are generating rules by some CRM algorithms and then fuzzifying the antecedent conditions. In the case of the third approach, simultaneously we are evolving MFs and classification rules. Let us discuss each of these approaches with their strength and weaknesses.

Unlike GAs for CRM whose antecedent conditions are Boolean, here the antecedent conditions are fuzzy. However, both the approaches are conceptually similar. Therefore, many aspects of GAs designed for classification rule generation can be used for FCR generation; in particular, the method of encoding an FCR antecedent into an individuals and determining the consequent part. The determination of the consequent part suggested in Ref 95 is as follows:

As described above, the fuzzy if–then rules are of following type: For \( M \)-class, \( d \)-dimensional classification problems, the fuzzy if–then rule is of the following form:

\[
\text{Rule } R : \text{IF } x_1 \text{ is } A_1, x_2 \text{ is } A_2, \ldots, x_d \text{ is } A_d \text{ THEN Class } C,
\]

where \( <x_1, x_2, \ldots, x_d> \) is a \( d \)-dimensional pattern vector, \( A_i \) is an antecedent fuzzy set, and \( C \) is the class label of the rule.

The consequent class \( C \) of each fuzzy rule is determined by the following widely used procedure proposed by Isibuchi and Nakashima et al.:

- Calculate the compatibility grade of each training pattern, \( \mu(x_p)(x_1, x_2, \ldots, x_d) \) with the fuzzy if–then rule. i.e., \( \mu(x_p) = \mu A_1(x_{p1}) \times \mu A_2(x_{p2}) \times \ldots \times \mu A_d(x_{pd}) \), where \( \mu A_i(x_{pi}) \) is the MF of the antecedent fuzzy set \( A_i \).
- Calculate the sum of the compatibility grades of the training patterns with the fuzzy if–then rule \( R \) for each class. i.e., \( \beta_{class}(R) = \sum_{x_p \in Classh} \mu(x_p), i = 1, 2, \ldots, M \).
- Find the class which has maximum \( \beta_{class}(R) \) value and denote it as \( \beta_{classh} \). If two or more classes take the maximum value, then there is no unique consequent class for rule \( R \) and \( C_i \) will be \( \phi \). If a single class takes the maximum value then \( C_i \) will be that class. Again if \( \beta_{class}(R) = 0 \), that is no training pattern is compatible with rule \( R \), then also the consequent class \( C_i \) will be \( \phi \).

The above procedure to choose the best rule consequent for a given rule antecedent seems intuitive and a generalization of CRM. However, in the case of fuzzy rules there can be a reasonable number of data instances whose degree of matching in the antecedent part can be close to 0; but their collective effort can have a significant influence of choosing a class. This leads to some undesirable results. In other words, it has a bias favoring the choice of majority class in the training set and this tends to be undesirable for the imbalanced class distribution data set. One solution to mitigate this problem is thresholding.

While computing the degree of matching in the antecedent part, the design should at least go for a data-sensitive operator, instead of adapting product operator as the only tool. An example of such an operator is the one based on the median of the
member degrees of all rule conditions, as proposed in Ref 165.

After generating the FCRs, we can now classify an unknown sample as follows:

A new input pattern \( x_\text{q} = < x_{q1}, x_{q2}, \ldots, x_{qd} > \) can be classified using any of the following two procedures:

1. Fuzzy reasoning based on a single winner rule: This is a two-step method described as follows:
   - Calculate \( \alpha_{\text{Class}} b = 1, 2, \ldots, M \) as \( \alpha_{\text{Class}} = \max (\mu (\tilde{x}_q)) \).
   - Classify \( \tilde{x}_q \) as the class with the maximum value of \( \alpha_{\text{Class}} \).

2. The fuzzy reasoning method based on voting by multiple fuzzy if–then rules: The class of an input pattern \( X_p \) based on voting by multiple fuzzy if–then rules that are compatible with \( X_p \), can be determined by the following two-step method as follows:
   - Calculate \( \alpha_{\text{Class}} b = 1, 2, \ldots, M \) as \( \alpha_{\text{Class}} = \sum_{R_i \in \mathcal{Q}, C_j = \text{Class}} R_i (X_p) \).
   - Classify \( X_p \) as the class with maximum value \( \alpha_{\text{Class}} \).

In contrast to the first approach, the second approach is only evolving the MFs. In other words, here the GAs are used for instance when crisp prediction rules were already discovered by a conventional rule induction algorithm and we just want to fuzzify the discovered crisp classification rules. One of the contributions in this direction is presented in Ref 166.

In the case of the third approach, i.e., GAs can also be used to evolve both the rules and MFs. In this case, the genotype of an individual has at least two kinds of genes: (i) genes defining contents of fuzzy rules, and (ii) genes defining MFs. Some of the representative contributions of this approach are presented in Refs 167, 168.

The fitness evaluation can be carried out by fuzzified confusion matrix. A taxonomy and current research trends and prospects of genetic fuzzy system is presented in Refs 170.

### GAs for Multiobjective FCRM

Traditionally, the main objective of fuzzy classification systems was to maximize the accuracy. The majority of the above-cited techniques are focused on the accuracy, neglecting the interpretability—the most distinguishable feature and the primary merit of FCR. Interpretability depends on several factors such as the number of features, fuzzy rules, antecedent fuzzy sets, shape of fuzzy sets, completeness, consistency, and compactness of fuzzy rules. Studies have addressed the issue of interpretability in FCR. These two objectives, accuracy of the system and interpretability of fuzzy rules are conflicting objectives and in practice one of these objectives prevails the other one. Therefore, recently researchers are trying to find a solution that can simultaneously optimize these objectives without prioritizing others. In Ref 182, Cordon presents a review on the most representative genetic fuzzy systems relying on Mamdani-type fuzzy rule based systems to search interpretable linguistic fuzzy models with a good accuracy.

FCRM is not only a multiobjective problem but also a NP-hard problem. For example, in the M-class, the \( d \)-dimensional classification problem, the fuzzy if–then rule is of the following form:

IF \( x_1 \) is \( A_1 \) and \( x_2 \) is \( A_2 \) and \ldots and \( x_d \) is \( A_d \) THEN class \( C \)

where \( \tilde{x} = < x_1, x_2, \ldots, x_d > \) is a \( d \)-dimensional pattern vector, \( A_i \) is an antecedent fuzzy set, and \( C \) is the class label of the rule.

Antecedents of fuzzy rules are \( k \) linguistic values and don’t care. So there are \((k + 1)^d \) fuzzy if–then rules, for the \( d \)-dimensional pattern classification problem. Hence the number of rules is very high. The primary goal in fuzzy rule generations is precision reduction and interpretability improvement.

Let us discuss the objectives of FCRM, which are normally considered in many objective optimizations.

The accuracy of fuzzy classification system is defined as

\[
J = 1/m \sum_{k=1}^{n} e_k, \tag{5}
\]

where \( e_k \) is the classification error for pattern \( \tilde{x}_k \) and is defined as \( e_k = 1 \), if \( \tilde{x}_k \) is classified correctly and \( e_k = 0 \), if \( \tilde{x}_k \) is classified falsely.

Interpretability refers to the ability to express the behavior of the system in a human understandable way. It is a subjective property, which has no formal definition but the following aspects are considered to describe this property:

- The number of variables: Fuzzy models should use fewer variables.
- The number of fuzzy rules: The fuzzy model should use very less fuzzy rules. As to human experience, this number should be less than 10.
• **Completeness, consistency, and compactness of fuzzy rules:** Fuzzy rules should cover the whole input space, i.e., for each effective input variable combination, there must be at least one fuzzy rule being fired. The rules in the rule base should not be contradictory to each other. There must be no rule, whose antecedent is a subset of another rule, and no rule may appear more than once in the rule base.

• **Characteristics of membership functions:** Normality and complexity are two essential properties of the MFs used in modeling.

An overview of interpretability measures is presented in Ref 188 in context of the linguistic fuzzy rule based system. More about interpretability can be obtained from Refs 172, 189, 190.

MOGAs are used to generate fuzzy rules with better trade-off between interpretability and accuracy. To achieve these objectives, different NSGA-II, SPEA, and SPEA2 algorithms are frequently used. Some of the modified versions are also proposed, as discussed below. Furthermore, multiobjective algorithms are used for feature selection, tuning of MFs in addition to rule selection. Some of representative studies are as follows:

Isibuchi et al.\textsuperscript{191} used a two-stage rule selection procedure. In the first step, a set of candidate fuzzy rules by the heuristic rule generation procedure is generated and then MOGAs are used for optimizing the criteria.

In 1995, Isibuchi et al.\textsuperscript{191} proposed a GA for two-objective fuzzy rule generation, where the two objectives are maximize the number of correctly classified training patterns and minimize the number of linguistic rules from the rule set. They express the two objectives into a single scaler function as

$$f(S) = W_{NCP} \times NCP(S) - W_S \times |S|,$$

where $W_{NCP}$ and $W_S$ are randomly specified weights, $S$ is the rule set, $NCP(S)$ is the number of correctly classified training patterns, and $|S|$ is the number of linguistic rules in $S$. For each pair of selected parent individuals, the weight $W_{NCP}$ is any random number from $[0, 1]$ and $W_S = 1 - W_{NCP}$. The multiple solutions are preserved from the current generation to the next generation as elite solutions. These elite solutions are randomly selected from a tentative set of nondominated solutions that are stored and updated at each generation of the two-objective GAs.

Furthermore, in 2001, Isibuchi and Yamamoto,\textsuperscript{154} proposed a three-objective GA to find nondominated rule sets. They considered the three objectives as

1. maximizing $f_1(S)$, the number of correctly classified training patterns,
2. minimizing $f_2(S)$, the number of fuzzy rules in $S$, and
3. minimizing $f_3(S)$, the total rule length of fuzzy rules in $S$,

where $S$ is the subset of generated fuzzy rule sets. A MOGA has been employed,\textsuperscript{191,192} which uses the fitness function as

$$fitness(S) = w_1 \times f_1(S) - w_2 \times f_2(S) - w_3 \times f_3(S),$$

where $w_1$, $w_2$, and $w_3$ are random weights satisfying the following conditions: $w_1, w_2, w_3 \geq 0$ and $w_1 + w_2 + w_3 = 1$. The nondominated rule sets are stored in a tentative pool, separated from current population. The pool is updated in every generation to store the nondominated rule sets, which are examined. From the pool, randomly $N_{elite}$ rule sets are selected as elite individuals and added to new solutions.

In 2004, Isibuchi et al.\textsuperscript{153} proposed a multiobjective genetic local search algorithm for generating a small number of fuzzy if–then rules for pattern classification. The method combines the local search and rule weight learning.\textsuperscript{154,195} In this framework, three objectives are used: maximization of classification accuracy, minimization of selected rules, and minimization of the total rule length. In the first stage, fuzzy if–then rules are generated and prescreened using two rule evaluation measures such as confidence and support used in data mining.\textsuperscript{3} The confidence and support of a fuzzy if–then rule $A_q \Rightarrow C_q$ is defined, respectively, as follows:

$$Confidence \ C(A_q \Rightarrow C_q) = \frac{|D(A_q) \cap D(C_q)|}{|D(A_q)|},$$

$$Support \ s(A_q \Rightarrow C_q) = \frac{|D(A_q) \cap D(C_q)|}{|D|},$$

where $D$ is the set of training patterns, $|D(A_q)|$ is the number of training patterns compatible with antecedent $A_q$, and $|D(A_q) \cap D(C_q)|$ is the number of training patterns that are compatible with both antecedent $A_q$ and consequent $C_q$. In the second stage, a number of nondominated rule sets with respect to the above objectives use a multiobjective genetic local search algorithm.

In 2006, they\textsuperscript{194,195} applied NSGA-II for multiobjective fuzzy rule selection with two heuristic tricks...
based on the problem, to find small rule sets with high accuracy. That is, they used the objectives in fuzzy rule selection, namely the minimization of the error rate on training patterns and the minimization of the number of fuzzy rules, in which biased mutation is used as the first trick. A kind of local search in removing the unnecessary rules is used as the second trick. These tricks are described as follows: As they used single winner rule-based methods for the classification of each pattern by the rule set $S$, some rules in this set do not qualify as a winner, for any rule. They eliminated these rules from the rule set $S$. This is performed after the first objective is calculated and before the second objective is calculated.

In 2006, Chen et al.\textsuperscript{196} proposed an approach based on MOGA to construct an interpretable and precision fuzzy classification from data. In the first stage, they used a MOGA for feature selection and dynamic grid partition with the following three objectives:

1. the number of wrongly classified training patterns $J_{err}$,
2. the number of features used $J_f$, and
3. the number of fuzzy rules $J_r$.

The fitness function is

$$\text{Min} F_1 = w_1 \times J_{err} + w_2 \times J_f + w_3 \times J_r,$$  \hspace{1cm} (10)

where $w_1$, $w_2$, and $w_3$ are positive weights as per the user’s choice. The number of rules $J_r$ is defined as $J_r = \prod_{j=1,...,n} k_j$, where $k_j$ is the number of dynamic grid partitions of feature $x_j$. Then they apply genetic operators to evolve the population. In the second step, they optimized the initial fuzzy classification systems so obtained in the above step. For this in the first step, they used a GA to exclude the unnecessary fuzzy rules and extracted the significant fuzzy rules. The objective is to select a subset of rules, keeping the classification performance. They used a single objective function

$$\text{Min} F_2 = w_4 \times J_{err} + w_5 \times J_r + w_6 \times J_m,$$  \hspace{1cm} (11)

where $J_{err}$ is the number of wrongly classified training patterns, $J_r$ is the number of fuzzy rules, and $J_m$ is the average length of fuzzy rules. $w_4$, $w_5$, and $w_6$ are positive weights. Again for improvement of classification performance of the fuzzy classification system, they used a GA to optimize the parameters. They form the chromosomes as a sequence of real numbers by coding the centers of MFs, the neighboring overlap values, and the certainty degrees of the consequents. They also restricted the search space of GA as the centers are limited in a range of $\pm \alpha$ % around their initial values; search spaces of the neighboring overlap values are constrained in $[0.02, 0.45]$ and certainty degree varies from 0 to 1.

In 2007, Alcala et al.\textsuperscript{197} proposed an accuracy-oriented multiobjective algorithm $\text{SEPA}_2^{\text{ACC}}$, based on the SPEA\textsuperscript{2} to obtain fuzzy rule-based systems with a better transaction between interpretability and accuracy in linguistic fuzzy modeling by performing a rule selection together with tuning of the MF, which minimizes only two objectives to achieve the desired trade-off: The number of rules and the mean squared error. $\text{SEPA}_2^{\text{ACC}}$ algorithm centers the search on the desired Pareto zone, i.e., the zone having high accuracy with least possible number of rules, by incorporating two changes in the SPEA2 algorithm. The objective is to put pressure on the selection of solutions with high accuracy. They have made two changes in the existing algorithms as follows:

1. A restarting operator is applied at Step 4 of the algorithm, by maintaining the most accurate individual as the single individual in the external population and obtaining the remaining individuals in the population with the same rule configuration of the best individual and tuning parameters generated at random within the corresponding variation intervals. Then return to Step 2 with $t = t + 2$.

2. As the second change, in each stage of the algorithm, before and after restarting, they sorted the solutions from the best to the worst considering accuracy as a sorting criterion. The number of solutions in the external population considered to form the mating pool should progressively reduce from 100\% at the beginning to 50\% at the end of each stage, by focusing only on those with the best accuracy. They have also modified the creation of solutions in the initial population. They selected all the possible rules that favor a progressive extraction of bad rules, only by means of mutation at the beginning and then by means of the crossover.

They used a double coding scheme for both rule selection $c_s$ and tuning $c_t$:

$$C = C_s^p, C_t^p.$$  \hspace{1cm} (12)

In the rule selection part, they used binary-coded strings with a size $m$, the number of initial rules. The corresponding gene is assigned as “1” if the rule is selected otherwise “0,” i.e., $C_s^p = (c_{s1}, c_{s2}, ..., c_{sm})$, where $c_{si} \in [0, 1]$. In the tuning part,
a real coding is considered, being $m'$, the number of labels of each of the $n$ variables comprising the database. $C_i = (a'_i, b'_1, c'_1, \ldots, a'_n, b'_m, c'_m)$, $i = 1, \ldots, n$ and $C^p_i = C_1 C_2 \ldots C_n$. In the $C_S$ part of the initial population, all individuals are selected with all genes having value “1” and in the $C_T$ part of initial population; The initial database is taken as a first individual. The rest of individuals are generated randomly within the corresponding variation intervals, which are calculated from the initial database. They applied the BLX-0.5 crossover operator in the $C_T$ part and HUX crossover in the $C_S$ part of the chromosome. Then finally four offspring are produced by combining two from the $C_S$ part and two from the $C_T$ part. They applied the mutation operator, which changes a gene value randomly selecting one in each $C_S$ and $C_T$ part with probability $P_m$.

In 2008, Gacto et al. proposed an extension of SEPA2$_{ACC}$ algorithm, SEPA2$_{ACC}^2$ to obtain the fuzzy rule based system with a better transaction between interpretability and accuracy in linguistic fuzzy modeling by performing rule selection together with a tuning of the MF, which minimizes only two objectives to achieve the desired trade-off: the number of rules and the mean squared error. They proposed two changes in the SEPA2$_{ACC}$ algorithm. Instead of HUX crossover in the $C_S$ part of the chromosome, an intelligent crossover is applied. Offspring are generated by applying the following steps:

- Apply BLX crossover to obtain the $C_T$ part of the offspring.
- In the $C_S$ part, for every gene, the corresponding rule is extracted from each individual in the crossover, namely offspring, parent 1, and parent 2, after the real parameters are obtained by determining a database. Similarly, the same rule is obtained three times with different MFs, which pertain to these three individuals.
- Considering only the center points of the MFs involved in the extracted rules, the Euclidean normalized distances are computed between offspring and each parent. The normalization of the differences between two points is done by the amplitude of their respective variation intervals.
- The present rule would be selected for the offspring or not is determined by the nearest parent, by copying its value in the $C_S$ part, for the corresponding gene.
- Until all the $C_S$ values are assigned for the offspring, repeat Step 1 through Step 4.

Using the above process, four offspring are generated. In real application, exploration is performed in the $C_T$ part using this operator and the $C_S$ part is obtained directly, based on the previous knowledge of parent’s regarding the inclusion or not inclusion of a specific configuration of MFs for every rule. When an offspring is generated, the mutation operator changes a gene value in the $C_T$ part randomly and in the $C_S$ part zero is set to a gene selected at random with probability $P_m$.

Pulkkinen and Koivisto have identified a fuzzy classifier by using decision tree and multiobjective evolutionary algorithms. To obtain the compact and accurate fuzzy classifier, Pulkkinen and coworkers presented a multiobjective genetic fuzzy systems in Ref 202. Marquez et al. have presented a multiobjective evolutionary algorithm with an interpretability improvement mechanism for linguistic fuzzy systems with adaptive defuzzification.

**SUMMARY AND FURTHER RESEARCH**

The motivation of applying evolutionary algorithms in KDD is that EAs are robust search methods, which perform a global search in the candidate solution space (feature space, rules, or another form of knowledge representation). Initially, we presented the preliminary concepts of KDD, EAs and its taxonomy, MOEAs, and fuzzy set theory. Then we discussed the significant advances of EAs, MOEAs, and fuzzy system in KDD and data mining area.

Furthermore, this paper has attempted to study about CRM using GAs. The reason for applying GAs for classification is that GAs can use the same knowledge representation (IF–THEN rules) as conventional rule induction algorithms. However, GAs’ global search nature tends to cope better with the attribute interaction and to discover interesting relationships that would be missed by greedy search of rule induction algorithms. The flexible algorithmic paradigm can be used to incorporate background knowledge into the GA and or to hybridize GAs with local search methods that are specifically tailored to the data mining tasks being solved.

Like any other data mining paradigm, GAs also have some disadvantages. One of them is that conventional genetic operators such as crossover and mutation operator are blind search operators in the sense that they modify individuals in a way independent from the individual fitness. This characteristic of conventional genetic operators increases the generality of GAs but intuitively tends to reduce their effectiveness in solving a specific kind of problem. Hence, it
is important to do more studies to extend GAs use to task-specific operators.

Another disadvantage of GAs is that they are computationally slow. However, if necessary, the processing time can be significantly reduced by using parallel processing techniques\textsuperscript{204,205} and/or compute the fitness of individuals by using only a subset of training instances. Another possibility is to compute the fitness of some of the individuals and approximate others. However, this needs intensive research.

An important research direction is to better exploit the power of GP, ES, EP, EDA, and CGAs in data mining. There are several GP algorithms for discovering classification rules\textsuperscript{206} or for classification in general\textsuperscript{207,208} However, the power of GP is still underexplored.

Furthermore, FCRM using is another attracting point of this paper. With a large number of criteria used to design a genetic fuzzy system considered by various researchers, the most frequently used ones are to (i) maximize the number of correctly classified training patterns, (ii) minimize the number of rules, and (iii) minimize the length of the fuzzy rule for finding out a genetic fuzzy classifier.

From this study, it is worth noting that the multiobjective nature of the fuzzy rule based systems needs intensive care for handling concavity of the problem and scalability when learning from large data sets. It is also equally important to consider the adaptation of MOGAs in FCRs to data set with a high imbalanced ratio.

Most of the studies only consider the quantitative measures of the fuzzy rule based systems and give less emphasis on qualitative measures. Hence, qualitative measures of fuzzy rule based systems still need further intensive research. In the context of the interpretability measure, our feature research includes designing of appropriate algorithms to handle the growing measures of interpretability within the framework of interpretability accuracy trade-off.

The hybridization between fuzzy systems and EAs in evolutionary fuzzy systems became an important research area during the past decade. Nowadays, it is a developed research area where researchers need to reflect to advance toward strengths and distinctive features of the fuzzy system. Hybridizing with other metaheuristics such as particle swarm optimization\textsuperscript{209–211} ant colony optimization\textsuperscript{212–214} and bee colony optimization\textsuperscript{215–217} can be another improvement in this direction.

NOTES

\textsuperscript{a}The problem of the rule interaction consists of evaluating the quality of a rule set as a whole, rather than just evaluating the quality of each rule in an isolated manner.

REFERENCES

29. Alberto P-M. Gene expression modular analysis an overview from the data mining perspective. WIREs Data Mining Knowl Discov 2011, 1:381–396.


134. Wu TP, Chen SM, A new method for constructing membership functions and fuzzy rules from training


Ishibuchi H, Murata T. A multi-objective genetic based local search algorithm and its application to...


