DOI: 10.1007/s11633-008-0022-2

Evolving Decision Rules to Predict Investment Opportunities

Alma Lilia Garcia-Almanza* Edward P. K. Tsang

Department of Computer Science, University of Essex, Wivenhoe Park, Colchester CO4 3SQ, UK

Abstract: This paper is motivated by the interest in finding significant movements in financial stock prices. However, when the number of profitable opportunities is scarce, the prediction of these cases is difficult. In a previous work, we have introduced evolving decision rules (EDR) to detect financial opportunities. The objective of EDR is to classify the minority class (positive cases) in imbalanced environments. EDR provides a range of classifications to find the best balance between not making mistakes and not missing opportunities. The goals of this paper are: 1) to show that EDR produces a range of solutions to suit the investor′s preferences and 2) to analyze the factors that benefit the performance of EDR. A series of experiments was performed. EDR was tested using a data set from the London Financial Market. To analyze the EDR behaviour, another experiment was carried out using three artificial data sets, whose solutions have different levels of complexity. Finally, an illustrative example was provided to show how a bigger collection of rules is able to classify more positive cases in imbalanced data sets. Experimental results show that: 1) EDR offers a range of solutions to fit the risk guidelines of different types of investors, and 2) a bigger collection of rules is able to classify more positive cases in imbalanced environments.

Keywords: Machine learning, genetic programming (GP), classification, imbalanced classes, evolution of rules.

1 Introduction

In a previous work of [1], we introduced an approach to generate and evolve a set of decision rules called evolving decision rules (EDR). The aims of EDR are as follows:

1) Classify the minority class in imbalanced environments;

2) Produce a range of solutions to suit different users preferences;

3) Generate comprehensive decision rules that can be understood by the user.

EDR is an evolutionary process that evolves decision rules; this is able to generate different solutions every time it is performed.

This paper is organized as follows. Section 2 exposes the motivation of this work and provides a brief explanation about previous works done in this area, disclosing the main differences among those works and our approach. Section 3 describes the procedure of EDR. Section 4 describes the experiments and results to test our approach. The conclusions are given in Section 5.

Notations

Rep	: Repository of rules.
μ	: Maximum number of rules in *Rep*.
φ	: Number of initial descendants per rule in *Rep*.
s	: Current number of rules in *Rep*.
ρ	: Size of the population.
β	: Percentage of population created by the rules.
h	: Hill-climbing probability.
τ	: A set of decision thresholds.
e_i	: Positive instance correctly predicted.

2 Motivation

The prediction of the minority class in imbalanced data sets is a problem in the machine learning field[2−5]. Unfortunately, many real world problems need the detection of rare cases, for example the detection of oil spillage[6], fraud detection[7], and illnesses prediction[8, 9].

To build an application that evolves and collects rules, we propose EDR, which is an evolutionary process that is based on selection, mutation, and hill-climbing. The aim of this method is to evolve a set of rules that holds different patterns of the positive cases. As shown in [10] and [11], some rules that classify positive cases could be eliminated during the evolutionary process. For that reason, we propose to collect rules to ensure that all the useful patterns produced by the evolutionary process will be included in the final solution of the problem. The result is a set of rules that are sorted according to their precision. The aim is to generate a range of solutions to suit different user′s preferences. Additionally, the analysis of the pattern(s) that are used to predict allows users to combine their knowledge to make a well-informed decision.

EDR[1] is inspired by two previous works: evolutionary dynamic data investment evaluator (EDDIE)[12−15] and repository method (RM)[10, 11, 16, 17]. EDDIE is a financial forecasting tool based on genetic programming (GP)[18]. EDDIE is trained using a set of examples to detect opportunities in future data sets. However, when the target is to detect high movements in stock prices, the number of positive examples is scarce and it becomes very difficult to identify those events.

On the other hand, RM analyzes the solutions proposed by a GP to gather rules (patterns) that classify the minority

Manuscript received August 20, 2007; revised October 30, 2007

*Corresponding author. E-mail address: algarc@essex.ac.uk class (positive cases) in diverse ways. The resulting patterns are used to classify the minority class in imbalanced data sets.

Previous works.

This section describes some works related to the evolution of decision rules. The majority of the works that evolve decision rules have used genetic algorithms to evolve a complete set of rules (Michigan approach[19]) or a population of set of rules (Pittsburgh approach[20, 21]).

Our approach has some similarities to the Michigan approach[19] because EDR evolves rules and the solution of the problem is composed by the complete set of rules. In contrast, in the Pittsburgh approach[20, 21], every individual in the population represents a complete set of rules. Thus, every individual holds an integral solution of the problem.

There are works that evolve decision rules by using a standard GP. Every decision tree classifies a single class. Once the evolutionary process has finished, the best individual of the evolution is converted into classification rules[22−24]. The corresponding authors claim that their approaches are able to generate comprehensible rules. However, GP tends to accumulate extra-code[18, 25, 26] and unless the rules are simplified, these can present the real variables and conditions that are involved in the rule. In contrast, EDR provides a simplification of rules that helps to identify the real conditions in the rules. In addition, to predict the minority class, especially in extreme imbalanced environments, it is better to collect all the available patterns. Otherwise, the best tree of the evolution would contain just patterns that may not repeat themselves in future data sets. The example in Section 4.3 illustrates that situation. None of the mentioned applications is able to provide a range of solutions to suit different user′s preferences.

3 Evolving decision rules procedure

This section describes the EDR procedure. It starts by describing the idea behind this approach, then an overview of EDR is provided and finally the description of each step is explained in detail.

EDR evolves a set of decision rules by using GP and it receives feedback from a key element that is called repository. Let us define the repository as a structure, whose objective is to store a set of rules. The resulting rules are used to create a range of classifications that allows the user to choose the best trade-off between the misclassifications and false alarms cost. Before explaining in detail how EDR works, let us give a general overview of the mentioned approach.

Step 1. Creation of the initial population. Create a population of random decision trees.

Step 2. Extraction of rules. Analyze every decision tree to delimit their rules and select the useful patterns.

Step 3. Rule simplification. Process the new rule to remove the redundant and vacuous conditions.

Step 4. Adding new rules to the repository. Detect new rules by comparing the new one with the existing rules in the repository. If the rule is different, then this is added to the repository. If there is a similar rule in the repository but the new one offers better performance then the old rule is replaced by the new one.

Step 5. Creation of a new population. Create the new population taking as parents the rules that are stored in the repository. The next generation will be created using the mutation operator and hill-climbing. The process is repeated from Step 2 until the algorithm has reached the maximum number of generations.

Step 6. Testing EDR. Once the evolutionary process has finished, EDR is tested by using another data set. It is evaluated by using sub-collections of rules from the repository. Those rules are grouped according to their precision (i.e., $Precision = \{1.00, 0.95, 0.90, \cdots, 0.05, 0.00\}$).

A general description of EDR has been introduced; in the following sections, this approach will be described in detail.

3.1 Initialization of population

The objective of this procedure is to create a collection of candidate solutions. We propose to create a population of decision trees using the discriminator grammar (DG) described in [1]. We used DG for the following reasons:

1) It simplifies the delimitations of the rules in the decision trees;

2) It maintains the valid structure of the individuals.

At this point, a question arises. If the system evolves decision rules, why is the population composed by decision trees? It is because a single individual (decision tree) could contain more than one rule, thus the creation of decision trees generates more solutions. Using disjunctions ("OR"), the number of rules in the decision tree increases notably; as a consequence, the probability of finding valuable patterns increases too.

3.2 Rule extraction

This part of the process analyzes every decision tree to delimit its rules and select those patterns that are useful to the classification. This process is applied to the entire population because it was showed that many useful rules can be obtained from the whole GP population[10, 27]. It was possible to discover predictive rules even in low-fitness individuals. The main reasons to divide a decision tree to its corresponding rules are as follows:

1) Identify the patterns in the decision trees;

2) Understand the rules easily.

Let T be a decision tree; the objective of this procedure is to find the set of rules, such as, $T = \{R_1 \vee R_2 \vee \cdots \vee R_n\}$. Let us define a rule $R_i \in T$ as a set of conditions that are associated by conjunctions "AND". R_i represents a minimal set of conditions that satisfies the tree T.

Once a rule $R_k \in T$ has been delimited, it is evaluated using the training data set. If the precision of R_k achieves a predefined precision threshold PT, where $PT > 0$, then R_k is considered for the Step 3 in Section 3, otherwise R_k is discarded.

3.3 Rule simplification

The aim of rule simplification is to remove vacuous and redundant conditions. A condition c_i in a rule R is redundant if there exists at least another condition $c_k \in R$ such as c_i implies c_k. In other words, redundant conditions are those which are repeated or report the same event e.g. $R_1 = \{var_1 > 0.5$ and $var_1 > 0.7\}$ the first condition is redundant. A condition c_i is vacuous in a rule R in relation to a data set D if under the conditions in R, c_i does not affect the decision of the rule, such as $Performance(R) = Performance(R - c_i)$. To simplify rules, we have divided the conditions to hard and flexible. Let a hard condition be the equation that compares two variables (e.g. $var_1 < var_2$). Let a flexible condition be the equation between a variable and a threshold (e.g. $var_1 < 0.8$). Let similar conditions be a group of conditions that have the same variable and operator. For example, $var_1 < 3$ and $var_1 < 2$ are similar conditions. Conditions have been divided to hard and flexible, because the conditions that compare thresholds could be difficult to differentiate (e.g. $var_1 < 0.8912$ and $var_1 < 0.8910$). However, these can be easily simplified (e.g. $var_1 < 0.8910$).

The simplification of rules is an important process for the following reasons:

1) Recognize the real variables and conditions that are involved in the rule;

2) Identify the duplication of rules in the repository. This assures the existence of different rules, increasing the variety of the solutions;

3) Remove the conditions that are not affecting the performance of the tree in the training data set. It reduces the risk of including conditions whose behavior could be unpredictable in future data sets.

3.4 Adding new rules in the repository

Once a rule R_k has been simplified, we have to determine the novelty of that rule by comparing R_k to the existing rules in the repository. To compare rules effectively, these have been divided in two categories, hard and flexible rules. Let R_i be a hard rule, if it is composed exclusively of hard conditions. Let R_i be a flexible rule if it has at least one flexible condition. Finally, let R_k and R_i be similar rules if these have the same hard conditions and similar flexible conditions. The comparison of hard rules is straightforward, but the comparison of flexible rules is more complex, because rules contain thresholds. The following policy determines the inclusion of rule R_k in *Rep*.

- If R_k is a hard rule and $\nexists\ R_i \in Rep$ such as $R_i = R_k$, then $Rep = Rep \cup R_k$, but if the cardinality of $|Rep| > \mu$, then $Rep = Rep - R_w$ where $R_w \in Rep$ and $Fitness(R_w) \leq Fitness(R_i)$, $\forall\ R_i \in Rep$.

- If R_k is a flexible rule and $\exists\ R_i \in Rep$ such as R_k and R_i are similar rules and $Fitness(R_k) > Fitness(R_i)$, then $Rep = (Rep - R_i) \cup R_k$.

- If R_k is a flexible rule and $\nexists\ R_i \in Rep$ such as R_k and R_i are similar rules, then $Rep = Rep \cup R_k$, but if the cardinality of $|Rep| > \mu$, then $Rep = Rep - R_w$ where $R_w \in Rep$ and $Fitness(R_w) \leq Fitness(R_i)\ \forall$ $R_i \in Rep$.

Notice that the number of rules is limited by the parameter μ and when the number of rules in *Rep* is bigger than μ, then the worse rule in *Rep* has to be deleted. The replacement of rules is an important part of this process because this is applied to flexible rules which hold conditions with continuous thresholds. Thus, every time a flexible rule is replaced by a better similar rule, the thresholds are being approximated to the "optimal" values. In this process, the performance is measured by the geometric mean of the product of precision times recall to encourage the recall.

3.5 Creation of a new population

This section describes the procedure to generate a new population of individuals. The population will be totally replaced by a new population of decision trees created by means of the mutation and hill-climbing of the existing rules in the repository. The number of rules in the repository is variable because it depends on the new patterns that have been found. The number of rules is limited by μ, which represents the maximum number of rules in the repository. The creation of a new generation follows the guidelines below.

- $(s \cdot \varphi \leq \rho)$. At the beginning of the evolutionary process, when the product of the current number of rules in *Rep* times the number of initial descendant per rule is less than the population size, the system will replace the population in the following generation with φ offspring per rule. If the number of new offspring is less than the population size, then the remaining individuals will be created randomly by the grow-method.

- $(s \cdot \varphi > \rho)$. It is obvious that the repository is continuously growing and so, there is a maximum number of rules that can be stored. Now, we have to consider when the product of the current number of rules in *Rep* multiplied by the number of initial descendant per rule is greater than the population size $(\varphi \cdot s > \rho)$, then the rules have to reproduce less and of course, the number of offspring is limited by $|\rho/s|$. Because the value of the division is truncated, the number of offspring is less than the size of the population, thus the remaining individuals are created at random.

- $(s = \mu$ and $s > \rho)$. When the repository is totally full and the number of rules is greater than the population size, just a fraction of rules in *Rep* is allowed to produce one descendant. The rules to produce offspring are selected randomly without any type of elitism. However, those descendants will produce only $\beta\%$ of the population. The remaining individuals will be created at random to create variety. β is a parameter, which is determined by the user.

- The hill-climbing is applied randomly using a probability h, which will be one of the descendants of a rule. The remaining individuals, if any, are produced by mutation.

3.6 Rule evaluation

Once the evolutionary process has finished, the final rules in the repository will be used to classify the testing data set as follows:

1) Sort the rules by precision in descending order;

2) Define a set of thresholds $\tau = \{\tau_i\}$ between $[0, 1]$ separated at regular intervals for example: τ=$\{0, 0.05, \cdots, 1\}$;

3) For each threshold τ_i, select those rules from *Rep* whose precision is greater or equal to τ_i, then store those rules in a sub-repository $Rep_{\tau_i} = \{R_{\tau_{ik}}\}$ where $R_{\tau_{ik}}$ is a rule, such as, $Precision(R_{\tau_{ik}}) \geq \tau_i \; \forall \; R_{\tau_{ik}} \in Rep_{\tau_i}$;

4) For each example in the data set, if at least one of the rules in the sub-repository satisfies the example, this is classified as positive, otherwise it is considered negative.

EDR has been designed to produce as many classifications as much as the number of thresholds τ. Each of those classifications can be plotted in the receiver operating characteristic (ROC) space[28, 29]. The result is a curve that can be used to measure the general performance of the classifier and to choose the best tradeoff between misclassifications and false alarms.

4 Experimental section

This section describes a series of experiments to test our approach. First, a series of experiments was performed to compare EDR performance with a standard GP and RM, as presented in Section 4.1. Next, an experiment to test the performance of RM in different levels of complexity is described in Section 4.2. Finally, Section 4.3 shows an example, which illustrates the role of a set of rules.

4.1 Experiment: Comparison of EDR with a standard GP and the RM

4.1.1 Objective

The objective of this experiment is to analyze the performance of EDR and to compare this with a standard GP and the RM performance.

4.1.2 Procedure

The set of rules generated by EDR was used to create twenty classifications using the same number of thresholds τ. The parameters used to run RM are listed in Fig. 1. The values of the corresponding parameters are listed in Table 1.

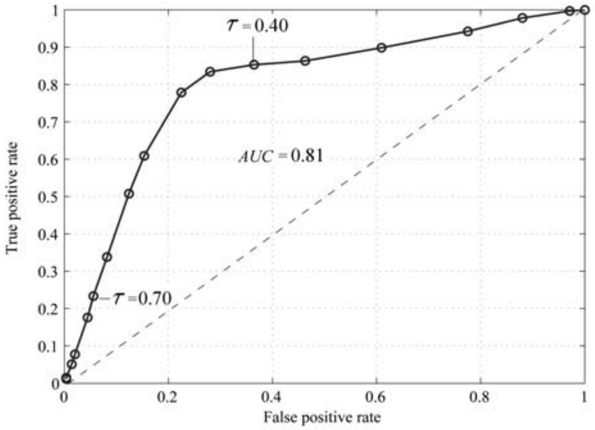

Fig. 1 Barclays′ parameters and results

Table 1 Barclays′ parameters (March, 1998 – January, 2005)

	Notations	Values
	Rate of return	15%
	Number of days	10
	Tendency movement	Increase
Training data set	Training examples	887
	Positive examples:	39 (4.3%)
Testing data set	Testing examples:	831
	Positive examples:	24 (2.8%)
EDR parameters	Number of runs	20
	Area under the ROC curve (AUC)	0.81
	Precision threshold (PT)	0.08
	Population size (ρ)	1000
	Number of initial spring (φ)	10
	Individuals random (β)	20
	Max number of rules (μ)	2500
	Number of generations	25
	Hill-climbing probability	0.14

4.1.3 Observations

Fig. 1 shows the ROC curve created by EDR. Notice that the classifications provided by EDR are well-distributed over the ROC curve, thus, it is possible to find conservative and liberal predictions. Now, let us discuss the points that form the curve. For instance, when τ is equal to 0.50, the recall has a high performance because EDR was able to detect 78% of the positive examples, this result has not sacrificed the accuracy (78%). When the investor′s risk-guidelines is conservative, EDR offers a range of suitable classifications, for instance, when the threshold τ= 0.70 the system is able to classify 23% (almost a quarter) of the positive cases with a very high accuracy (92%). On the other hand, when τ is smaller than or equal to 0.40, then the classifier′s performance tends to decrease because the number of new positive cases that are detected is paid with a serious decrease on the accuracy and precision. Table 2 shows the results using Barclays′ parameters.

Table 2 EDR results using Barclays′ parameters

τ	*Recall*	*Precision*	*Accuracy*	τ	*Recall*	*Precision*	*Accuracy*
1.00	0.01	0.09	0.97	0.50	0.78	0.10	0.78
0.95	0.01	0.09	0.97	0.45	0.83	0.08	0.72
0.90	0.02	0.11	0.97	0.40	0.85	0.07	0.64
0.85	0.05	0.10	0.96	0.35	0.86	0.05	0.55
0.80	0.08	0.10	0.95	0.30	0.90	0.04	0.41
0.75	0.18	0.11	0.93	0.25	0.94	0.04	0.25
0.70	0.23	0.11	0.92	0.20	0.98	0.03	0.14
0.65	0.34	0.11	0.90	0.15	1.00	0.03	0.06
0.60	0.51	0.11	0.87	0.10	1.00	0.03	0.06
0.55	0.61	0.11	0.84	0.05	1.00	0.03	0.06

Standard GP comparison. The result of the best individual produced by a standard GP is *Recall* = 0.14, *Precision* = 0.04, and *Accuracy* = 0.87. This result is plotted in the point (0.10, 0.15) in the ROC space, which describes a conservative prediction. As can be noticed, a standard GP produces a single prediction for every data set. In contrast, EDR provides a range of classifications, which allows the investors to tune the prediction according to their risk guidelines. If the requirement of the user is to detect as many positive cases as possible, τ has to decrease to move to the liberal part in the ROC space. In contrast, if the user′s preference is to reduce the risk, then τ has to increase to move to the conservative part in the graph. It is noted that the recall declines when τ increases because the selection of rules becomes stricter and fewer rules are selected, decreasing the number and variety of the rules in the repository. Note that all the results provided by EDR have better recall than the recall of the best GP individual. The result of the standard GP is plotted in (0.10, 0.15) in the ROC space which describes a conservative prediction. The EDR predictions have been distributed along the ROC curve. This allows investors to choose among a range of options the most suitable prediction for their requirements. According to the experimental results, it was possible to detect from 23% to 78% of the positive cases with an accuracy higher than 78%. The GP has better accuracy than the majority of the choices in EDR because the GP tends to predict negative classification, which has a high chance of being correct. The experiment shows that EDR is able to pick out rules that together classify more positive cases.

RM comparison. The result provided by RM and EDR is a set of classifications, which are distributed in the ROC space. Thus, we use the area under the ROC curve (*AUC*) to compare the performance of the classifiers. *AUC* obtained by RM is 0.77, which is outperformed by *AUC* generated by EDR (0.81).

4.2 Experiment to test different levels of complexity

4.2.1 Objective

The objective of this experiment is to test EDR using two data sets with different levels of complexity and a data set whose signal was labeled at random, which means that it does not have any pattern.

4.2.2 Procedure

We do not have a formal definition about rule complexity. However, for simplicity we measure the complexity of the data sets for the number of conditions involved in the solution. When the number of conditions in the solution increases, the solution is more complex. To control the complexity of the data sets in the experiment, we created three artificial data sets. The data set $Artificial_1$ was generated as follows:

Step 1. A set of 1 200 records was generated, every record held eight independent variables with real values. Every variable was randomly generated in a range of [0, 1].

Step 2. Every record was labeled with a class (positive or negative). The records that meet the requirements in at least one of the rules in S_1, as shown in (1) below, is labeled as positive, otherwise the record is classified as negative.

Step 3. The data was split in two data sets (training and testing) holding the same number of records (600).

The second artificial data set $Artificial_2$ was created repeating the above Steps 1–3 , but using S_2 instead of S_1. And the third data set ($Artificial_3$) was created using S_3 in (1).

$$\begin{array}{llll}
S_1 & = & \{ & R_1 = var_1 > 0.99 \\
 & & & R_2 = var_2 < 0.009 \\
 & & & R_3 = var_5 < 0.898 \text{ and } var_5 > 0.89 \\
 & & & R_4 = var_5 < 0.01 \\
 & & & R_5 = var_6 > 0.88 \text{ and } var_6 < 0.89 \ \} \\
 & & & \\
S_2 & = & \{ & R_1 = var_1 > 0.5 \text{ and } var_1 < 0.58 \text{ and } var_2 > 0.5 \text{ and } var_3 < 0.7 \text{ and } var_4 < var_3 \\
 & & & R_2 = var_3 < 0.45 \text{ and } var_3 > var_2 \text{ and } var_3 > var_4 \text{ and } var_3 > var_5 \text{ and } var_3 > var_6 \\
 & & & R_3 = var_8 < 0.898 \text{ and } var_8 > 0.86 \text{ and } var_5 > 0.065 \text{ and } var_5 < 0.35 \text{ and } var_3 > var_7 \\
 & & & R_4 = var_1 > 0.5 \text{ and } var_1 < 0.58 \text{ and } var_2 > 0.5 \text{ and } var_3 < 0.7 \text{ and } var_4 < var_3 \text{ and } var_4 < var_6 \\
 & & & R_5 = var_6 > 0.56 \text{ and } var_7 > var_6 \text{ and } var_8 > var_6 \text{ and } var_8 < var_1 \\
 & & & R_6 = var_1 > var_7 \text{ and } var_1 > var_6 \text{ and } var_6 < 0.23 \text{ and } var_5 < var_6 \} \\
 & & & \\
S_3 & = & & \text{Random selection}
\end{array} \qquad (1)$$

4.2.3 Observation

Before starting the analysis of the results in the testing data sets, let us present the results obtained by EDR in the training data set for each of the experiments. Using $Artificial_1$ AUC = 0.99, $Artificial_2$ AUC = 0.90 and $Artificial_3$ AUC = 0.91. Fig. 2 shows the results using three artificial data sets. The values of the corresponding artificial data set are listed in Table 3. As can be observed in all cases, EDR captured patterns from the training data sets. However, let us analyze the results in the testing data sets.

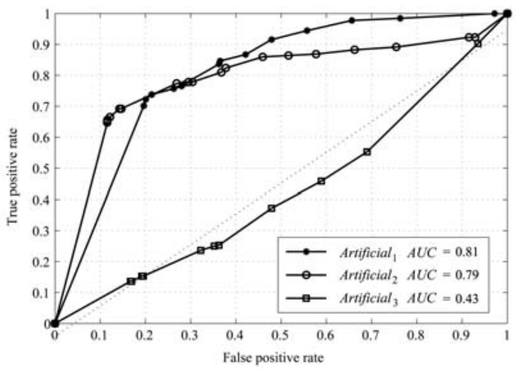

Fig. 2 Results using three artificial data sets (easy, difficult, and random)

Table 3 The values of artificial data sets

	Notations	Values
Training data set	Number of examples :	600
	Positive examples ($Artificial_1$)	28 (4.6%)
	Positive examples ($Artificial_2$)	18 (3%)
	Positive examples ($Artificial_3$)	18 (3%)
Testing data set	Number of examples	600
	Positive examples ($Artificial_1$)	29 (4.8%)
	Positive examples ($Artificial_2$)	29 (2.8%)
	Positive examples ($Artificial_3$)	18 (3%)
	Number of runs	20
	$Artificial_1$ AUC	0.81
	$Artificial_2$ AUC	0.79
	$Artificial_3$ AUC	0.43
EDR parameters	Precision threshold (PT)	6%
	Population size (ρ)	1000
	Number of initial spring (φ)	10
	Individuals at random (β)	15%
	Max number of rules (μ)	2000
	Number of generations	30
	Hill-climbing probability	15%

$Artificial_1$**.** According to our definition of complexity, $Artificial_1$ is a data set that has a low level of complexity. As can be observed from Fig. 2, AUC obtained by EDR is 0.81. Surprisingly, EDR does not offer any conservative prediction as can be noticed in Table 4. However, EDR found classifications that detect 70% of the positive cases with accuracy of 80% when $\tau > 0.85$. On the other hand, when τ decreases, the detection of positive cases increases steadily as the number of false alarms.

Table 4 EDR results using $Artificial_1$ data set

τ	*Recall*	*Precision*	*Accuracy*	τ	*Recall*	*Precision*	*Accuracy*
1.00	0.70	0.15	0.80	0.50	0.84	0.10	0.65
0.95	0.70	0.15	0.80	0.45	0.85	0.11	0.65
0.90	0.70	0.15	0.80	0.40	0.87	0.09	0.59
0.85	0.70	0.15	0.80	0.35	0.92	0.09	0.54
0.80	0.72	0.15	0.80	0.30	0.94	0.08	0.47
0.75	0.74	0.15	0.78	0.25	0.98	0.07	0.37
0.70	0.74	0.15	0.78	0.20	0.98	0.06	0.27
0.65	0.76	0.13	0.74	0.15	1.00	0.05	0.08
0.60	0.77	0.12	0.72	0.10	1.00	0.05	0.05
0.55	0.77	0.12	0.72	0.05	1.00	0.05	0.05

$Artificial_2$**.** $Artificial_2$ is more complex than the previous data set. As can be observed from Fig. 2, AUC obtained by EDR is 0.79. It means that the performance of EDR in $Artificial_1$ was slightly better than in $Artificial_2$. As can be observed in the ROC curve, EDR was unable to classify a small part of the positive cases. Those cases were classified just when τ was really low. Table 5 shows the results using $Artificial_2$ data set. To explain this phenomenon, the data set was analyzed, and it was discovered that rule $R_3 \in S_2$ produces a single positive case in the testing data set but it did not generate any positive case in the training data set. This means that the instance was not identified because there was no pattern to train EDR.

Table 5 EDR results using $Artificial_2$ data set

τ	*Recall*	*Precision*	*Accuracy*	τ	*Recall*	*Precision*	*Accuracy*
1.00	0.65	0.14	0.88	0.50	0.81	0.06	0.64
0.95	0.65	0.14	0.88	0.45	0.82	0.06	0.63
0.90	0.65	0.14	0.88	0.40	0.86	0.05	0.55
0.85	0.66	0.14	0.88	0.35	0.86	0.05	0.49
0.80	0.67	0.14	0.87	0.30	0.87	0.04	0.44
0.75	0.69	0.12	0.85	0.25	0.88	0.04	0.35
0.70	0.69	0.12	0.85	0.20	0.89	0.03	0.26
0.65	0.77	0.08	0.73	0.15	0.92	0.03	0.11
0.60	0.78	0.07	0.71	0.10	0.92	0.03	0.10
0.55	0.78	0.07	0.70	0.05	0.92	0.03	0.10

$Artificial_3$**.** As was explained previously, this data set was labeled randomly. This means that there are no patterns in the training data set to identify similar cases in future data sets. As was expected, EDR gathered patterns from the training data set, but these were not repeated in the testing data set. Table 6 shows the results using $Artificial_3$ data set. Fig. 2 shows the ROC curve plotted by EDR using $Artificial_3$. As can be seen, AUC obtained is 0.43, the performance of EDR was very low. EDR produced a random classification because the patterns in the training data set were not representative of the examples in the testing data set. The main reasons to get a low performance in a classifier based on supervised learning can be summarized as follows:

1) The data set does not contain any patterns or the independent variables do not describe the behavior of the data set;

2) The signal in the data set is labeled incorrectly;

3) The patterns in the training data set do not repeat in the testing data set.

The general observation of this experiment is that EDR is able to discover patterns to classify rare cases in imbalanced data sets. However, it is necessary to provide a representative training data set to capture the patterns to predict future cases. The complexity of the rules does not seem to affect seriously the performance of EDR. However, more research needs to be done about this.

Table 6 EDR results using $Artificial_3$ data set

τ	*Recall*	*Precision*	*Accuracy*	τ	*Recall*	*Precision*	*Accuracy*
1.00	0.14	0.02	0.81	0.50	0.24	0.02	0.66
0.95	0.14	0.02	0.81	0.45	0.24	0.02	0.66
0.90	0.14	0.02	0.81	0.40	0.25	0.02	0.64
0.85	0.14	0.02	0.81	0.35	0.25	0.02	0.63
0.80	0.14	0.02	0.81	0.30	0.37	0.02	0.52
0.75	0.14	0.02	0.81	0.25	0.46	0.02	0.41
0.70	0.14	0.02	0.81	0.20	0.55	0.02	0.32
0.65	0.15	0.02	0.79	0.15	0.90	0.03	0.09
0.60	0.15	0.02	0.79	0.10	0.90	0.03	0.09
0.55	0.15	0.02	0.79	0.05	0.90	0.03	0.09

4.3 An illustrative example to analyze a set of decision rules produced by EDR

4.3.1 Objective

This section analyzes a set of decision rules that were produced by EDR. The objective of this study is to show how a bigger collection of rules can help to detect the minority class in imbalanced environments.

4.3.2 Procedure

The example was taken from Barclays, where the training and the testing data sets are composed by 400 records each. The training data set has 15 positive examples, while the testing has 13. The set of rules for this analysis achieved a precision of 1 in the training data set, the set of rules is displayed in Table 7. As can be seen from Table 8, the complete set of rules classifies eight positive instances (*Recall* = 53%) in the training data set. As can be observed, every rule classifies three or four instances each. Given that the precision of the rules was 1, it means that the rules do not classify any negative case. Obviously, there is overlapping in the classification. However, an important question arises here. Is it useful to keep a collection of rules that overlap their predictions? This question is relevant because this will support one of the main claims of this method.

4.3.3 Observations

Let us analyze the set of rules in Table 7. Because the novelty is a basic condition to be included in the repository, notice that rules contain common conditions, but there are no identical rules. However, there is overlapping in the classification and even identical classification in rules R_6, R_7, R_8, R_9 and in rules R_{11}, R_{14}, R_{15}, and R_{17}. It means that the genotype is different, but the phenotype is similar. Obviously, we are taking as the phenotype the behavior of the rule. In the mentioned cases, the rules produced the same results. Let us analyze in detail the conditions and variables that are involved in each set of rules.

Table 7 Set of rules for the example in Section 4.3

Rule	Rule description	Detections
R_1	$var_9 > var_{15}$ and $var_{10} < var_{17}$ and $var_7 > 0.0727$	4
R_2	$var_3 > var_{24}$ and $var_6 < var_7$ and $var_{21} > var_{24}$	4
R_3	$var_3 > var_{20}$ and $var_{13} < -527.9$	4
R_4	$var_3 > var_{18}$ and $var_6 > var_{13}$ and $var_9 < var_{18}$ and $var_{21} > var_{24}$	4
R_5	$var_3 > var_{24}$ and $var_6 < 0.0413$	4
R_6	$var_3 > var_{18}$ and $var_{11} < -0.5056$	4
R_7	$var_4 > var_{15}$ and $var_{10} < var_{17}$ and $var_7 > 0.0727$	4
R_8	$var_{10} < var_{17}$ and $var_{15} < var_{22}$ and $var_7 > 0.0727$	4
R_9	$var_{10} < var_{17}$ and $var_{15} < var_{20}$ and $var_7 > 0.0727$	4
R_{10}	$var_1 < var_{21}$ and $var_2 < var_{21}$ and $var_3 > var_{21}$ and $var_{24} < 0.0136$	3
R_{11}	$var_8 > var_{13}$ and $var_7 > 0.0727$ and $var_{12} > -1082.$	3
R_{12}	$var_{10} < var_{20}$ and $var_{15} < var_{23}$ and $var_7 > 0.0727$	3
R_{13}	$var_{10} < var_{21}$ and $var_6 < 0.0936$ and $var_{22} < -0.067$	3
R_{14}	$var_7 > 0.0727$ and $var_{12} > -1082.$ and $var_{13} < -0.160$	3
R_{15}	$var_4 < -0.213$ and $var_7 > 0.0727$ and $var_{12} > -1185.$	3
R_{16}	$var_{10} < var_{20}$ and $var_{15} < var_{20}$ and $var_7 > 0.0727$	3
R_{17}	$var_1 < -0.072$ and $var_5 > 0.0445$	3

Table 8 Positive instances classified in the training data set

	Positive instance correctly predicted								
Rule	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	Sum
R_1		X	X			X	X		4
R_2				X	X		X	X	4
R_3			X	X	X			X	4
R_4					X	X	X	X	4
R_5				X	X		X	X	4
R_6		X	X			X	X		4
R_7		X	X			X	X		4
R_8		X	X			X	X		4
R_9		X	X			X	X		4
R_{10}		X	X		X				3
R_{11}	X	X	X						3
R_{12}		X	X			X			3
R_{13}			X			X	X		3
R_{14}	X	X	X						3
R_{15}	X	X	X						3
R_{16}		X	X			X			3
R_{17}	X	X	X						3
Sum	4	12	14	3	5	9	9	4	

Let $S_a = \{R_6, R_7, R_8, R_9\}$ be the set of rules. As was expected, S_a holds a set of different rules because EDR provides a mechanism to select different patterns, avoiding repeating the same rules in the repository (see Section 3.2). However, it is important to analyze the variables and relations that are involved in each rule in order to determine if those rules could be correlated.

As can be seen from Table 7, R_6 is different from the other rules in S_a, because R_6 does not have any equal hard condition or similar condition with another rule in S_a. On the other hand, R_7, R_8, and R_9 share the conditions: $var_{10} < var_{17}$ and $var_7 > 0.0727$, as the following paragraph shows

$$R_6 = \quad var_3 > var_{18} \wedge var_{11} < -0.5056$$

$$\begin{aligned} R_7 &= var_4 > var_{15} \wedge \quad \overbrace{var_{10} < var_{17} \wedge var_7 > 0.0727}^{\text{Common conditions}} \\ R_8 &= var_{15} < var_{22} \wedge \quad var_{10} < var_{17} \wedge var_7 > 0.0727 \\ R_9 &= \underbrace{var_{15} < var_{20}}_{\text{Different condition}} \wedge \quad var_{10} < var_{17} \wedge var_7 > 0.0727\ . \end{aligned}$$

Let R_c be the rule that is formed by the common conditions in R_7, R_8 and R_9 thus $R_c = \{var_{10} < var_{17} \wedge var_7 > 0.0727\}$. A new evaluation was performed using R_c, the result was $TP = 4$, $FP = 15$, $FN = 11$, and $TN = 370$, where TP, FP, FN, and TN denote true positive, false positive, false negative, and true negative, respectively. It means that rules R_7, R_8 and R_9 are more specialized than R_c, because R_c classifies 15 false alarms.

Given that var_{15} is involved in different conditions in R_7, R_8, R_9, it is important to verify if the other variables (var_{22} and var_{20}) are correlated and if var_4 is inversely correlated to var_{22} and/or var_{20}. As can be seen from Table 9, the indicators are: var_4 = Price trading breaking rule 50 days, var_{20} = LIBOR: 3 months moving average 50 days, and var_{22} = UK01Y00MA moving average 50 days. At this point, the financial and technical analysis knowledge of the user is crucial to determine the conditions in the rules. Now, let us analyze the following set of rules: $S_b = \{R_{11}, R_{14}, R_{15}, R_{17}\}$.

Obviously, S_b holds a set of different rules. However, it is important to analyse the variables and relations that are involved in each rule to determine if those rules could be correlated. As can be seen from Table 7, R_{17} is different from the other rules in S_b. This is because R_{17} does not have any equal hard condition or similar condition with another rule in S_b. On the other hand, R_{11}, R_{14}, and R_{15} have in common the conditions: $var_7 < 0.0727$ and $var_{12} > Threshold$, where $Threshold \in [-1185, -1082]$, as shown below.

$$R_{17} = \quad var_1 < -0.072 \wedge var_5 > -0.0445$$

$$\begin{aligned} R_{11} &= var_8 < var_{13} \wedge \overbrace{var_7 > 0.0727 \wedge var_{12} > -1082}^{\text{Common conditions}} \\ R_{14} &= var_{13} < -0.16 \wedge var_7 > 0.0727 \wedge var_{12} > -1082 \\ R_{15} &= \underbrace{var_4 < -0.213}_{\text{Different condition}} \wedge var_7 > 0.0727 \wedge var_{12} > -1185. \end{aligned}$$

Let R_d be the rule that is formed by the common conditions in R_{11}, R_{14}, and R_{15} thus $R_c = \{var_7 > 0.0727 \wedge var_{12} > -1185\}$. A new evaluation was performed using R_d, the result was $TP = 3$, $FP = 11$, $FN = 12$, and $TN = 374$. It means that R_{11}, R_{14}, and R_{15} are more specialized than R_d because these do not classify any false alarm as R_d does. R_{11} differs because of the condition $var_8 < var_{13}$ where price volatility of 50 days is bigger than moving average of 10 days of the momentum indicator of 10 days. The fact that R_{11} and R_{14} classify the same instances in the testing data set could suggest a correlation between them. Finally, it is important to verify if var_4 and var_{13} are correlated. As can be seen in Table 9, the indicators are: var_4 = Price trading breaking rule 50 days, var_{12} = Momentum indicator 60 days.

Table 9 The values of variables

Variables	Values
var_1	Price moving average 12 days
var_2	Price moving average 12 days
var_3	Price trading breaking rule 5 days
var_4	Price trading breaking rule 50 days
var_5	Filter rule 5 days
var_6	Filter rule 63 days
var_7	Price volatility 12 days
var_8	Price volatility 50 days
var_9	Volume moving average 10 days
var_{10}	Volume moving average 60 days
var_{11}	Momentum indicator 10 days
var_{12}	Momentum indicator 60 days
var_{13}	Momentum 10 days moving average 10 days
var_{14}	Momentum 60 days moving average 60 days
var_{15}	Generalized momentum indicator 10 days
var_{16}	Generalized momentum indicator 60 days
var_{17}	FOOTSIE moving average 12 days
var_{18}	FOOTSIE moving average 50 days
var_{19}	LIBOR: 3 months moving average 12 days
var_{20}	LIBOR: 3 months moving average 50 days
var_{21}	UK01Y00 moving average 12 days
var_{22}	UK01Y00MA moving average 50 days
var_{23}	UK10Y00MA moving average 12 days
var_{24}	UK10Y00MA moving average 50 days

There is a great variety of decision trees that can be formed with a subset of rules in Table 7 that does not classify any positive case in the testing data set, for example:

$$T_a = \{R_2, R_4, R_{17}\}$$

$$T_b = \{R_1, R_5, R_{15}\}$$

$$T_c = \{R_3, R_4, R_{11}\}.$$

As can be seen in Table 10, T_a, T_b, and T_c are not able to classify any positive case in the testing data set. However, if we evaluate the fitness of T_a, T_b, and T_c using the training data set, the performance is equal to the complete set of rules in Table 7, because every rule has $Precision = 1$. It means that the classification in the training data set for: $T_a = T_b = T_c = \{TP = 8, FP = 0, FN = 5$ and $TN =$

387}. However, in the testing data set, T_a, T_b, and T_c do not classify any positive example. There are many classifier systems based on GP which claim to evolve a sets of rules. These calculate the fitness of the individual by measuring the result of the classification and the "simplicity" of the solutions, as an instance[23, 24, 30, 31]. Other works just divide the resulting decision tree in rules, for example [22]. In those cases the GP is favoring the shortest solutions as T_a, T_b, and T_c instead of a bigger tree that hold more rules. On the other hand Yin et al.[32] create a set of rules using GP discarding rules to find the minimal set of rules. That procedure was implemented to reduce the bloat in the evolutionary process.

Table 10 Positive instances classified in the testing data set

	Positive instance correctly predicted			
Rule	e_1	e_2	e_3	Sum
R_1				0
R_2				0
R_3				0
R_4				0
R_5				0
R_6		X		1
R_7				0
R_8				0
R_9				0
R_{10}				0
R_{11}	X			1
R_{12}				0
R_{13}			X	1
R_{14}	X			1
R_{15}				0
R_{16}				0
R_{17}				0
Sum	2	1	1	

5 Conclusions

In this paper, we have explained the EDR approach. This method was designed to classify the minority class in imbalanced data sets. The system′s output is a set of decision rules, which based on a threshold τ produces a range of classifications to suit the investor's preferences. For detailed analysis, we have used the ROC curve, which has helped to visualize the distribution of the classifications in the ROC space. In the same vein, we have used the area under the ROC curve (AUC) to measure the general performance of our approach and to compare this with the RM proposed in [10, 16, 27] and a standard GP.

The core of our approach is based on GP, which is aided by a repository of rules. The aim of this repository is to collect useful patterns that are used to produce the following population in the evolutionary process. The main operators of EDR are the mutation and hill-climbing, which produce instances of the collected patterns. Furthermore, a simplification process is performed to simplify the rules in the repository in order to produce understandable solutions. On the other hand, the removal of extra-code allows us to decrease the computational effort.

From experimental results in Section 4.1, it was observed that EDR produces a series of classifications to adapt to the user needs (from conservative to liberal predictions).

An illustrative example was analyzed in Section 4.3 to explain how a bigger collection of rules is able to classify more positive cases in imbalanced environments. As it can be observed from this example, EDR produces comprehensible rules that can be analyzed by the user to understand the conditions and variables in the rule. Thus, the users can combine their knowledge to make a more informed decision.

References

[1] A. L. Garcia-Almanza, E. P. K. Tsang, E. Galvan-Lopez. Evolving Decision Rules to Discover Patterns in Financial Data Sets. *Computational Methods in Financial Engineering*, to be published.

[2] N. Japkowicz, The Class Imbalance Problem: Significance and Strategies. In *Proceedings of International Conference on Artificial Intelligence*, Las Vegas, Nevada, USA, vol. 1, pp. 111–117, 2000.

[3] G. M. Weiss. Mining with Rarity: A Unifying Framework. *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 7–19, 2004.

[4] G. E. A. P. A. Batista, R. C. Prati, M. C. Monard. A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data. *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 20–29, 2004.

[5] K. McCarthy, B. Zabar, G. Weiss. Does Cost-sensitive Learning Beat Sampling for Classifying Rare Classes? In *Proceedings of the 1st International Workshop on Utility-based Data Mining*, ACM Press, New York, NY, USA, pp. 69–77, 2005.

[6] M. Kubat, R. C. Holte, S. Matwin. Machine Learning for the Detection of Oil Spills in Satellite Radar Images. *Machine Learning*, vol. 30, no. 2-3, pp. 195–215, 1998.

[7] T. Fawcett, F. J. Provost. Adaptive Fraud Detection. *Data Mining and Knowledge Discovery*, vol. 1, no. 3, pp. 291–316, 1997.

[8] M. Greiner, D. Pfeiffer, R. D. Smith. Principles and Practical Application of Receiver-operating Characteristic Analysis for Diagnostic Tests, *Preventive Veterinary Medicine*, vol. 45, no. 1-2, pp. 23–41, 2000.

[9] G. Vanagas. Receiver Operating Characteristic Curves and Comparison of Cardiac Surgery Risk Stratification Systems. *Interactive Cardiovascular and Thoracic Surgery*, vol. 3, no. 2, pp. 319–322, 2004.

[10] A. L. Garcia-Almanza, E. P. K. Tsang. Forecasting Stock Prices Using Genetic Programming and Chance Discovery. In *Proceedings of the 12th International Conference on Computing in Economics and Finance*, 2006, [Online], Available: http://ideas.repec.org/p/sce/scecfa/489.html, July 4, 2006.

[11] A. L. Garcia-Almanza, E. P. K. Tsang. Detection of Stock Price Movements Using Chance Discovery and Genetic Programming. *Innovation in Knowledge-based and Intelligent Engineering Systems*, to be published.

[12] E. P. K. Tsang, J. Li, J. M. Butler. Eddie Beats the Bookies. *Software: Practice and Experience*, vol. 28, no. 10, pp. 1033–1043, 1998.

[13] J. Li, A Genetic Programming Based Tool for Financial Forecasting, Ph. D. dessertation, Department of Computer Science, University of Essex, UK, 2001.

[14] E. P. K. Tsang, P. Yung, J. Li. Eddie-automation: A Decision Support Tool for Financial Forecasting. *Journal of Decision Support Systems*, vol. 37, no. 4, pp. 559–565, 2004.

[15] E. P. K. Tsang, S. Markose, H. Er. Chance Discovery in Stock Index Option and Future Arbitrage. *New Mathematics and Natural Computation*, vol. 1, no. 3, pp. 435–447, 2005.

[16] A. L. Garcia-Almanza, E. P. K. Tsang. The Repository Method for Chance Discovery in Financial Forecasting. In *Proceedings of the 10th International Conference on Knowledge-based and Intelligent Information and Engineering Systems, Lecture Notes in Computer Science*, Springer-Verlag, Boumemouth, UK, pp. 30–37, 2006.

[17] A. L. Garcia-Almanza, E. P. K. Tsang. Repository Method to Suit Different Investment Strategies. In *Proceedings of IEEE Congress on Evolutionary Computation*, Singapore, pp. 790–797, 2007.

[18] J. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge, Massachusetts, UK, 1992.

[19] L. J. Fogel, A. J. Owens, M. J. Walsh. *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI, USA, pp. 131–156, 1975.

[20] S. Smith. Flexible Learning of Problem Solving Heuristics through Adaptive Search. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, Karlsruhe, FRG, pp. 422–425, 1983.

[21] K. A. De Jong, W. M. Spears. Learning Concept Classification Rules using Genetic Algorithms. In *Proceedings of the 12th International Conference on Artificial Intelligence*, Sidney, Australia, pp. 651–656, 1991.

[22] A. Niimi, E. Tazaki. Rule Discovery Technique Using Genetic Programming Combined with Apriori Algorithm. In *Proceedings of the 3rd International Conference on Discovery Science, Lecture Notes in Computer Science*, Springer, Kyoto, Japan, vol. 1967, pp. 273–277, 2000.

[23] C. C. Bojarczuk, H. S. Lopes, A. A. Freitas. An Innovative Application of a Constrained-syntax Genetic Programming System to the Problem of Predicting Survival of Patients. In *Proceedings of the 6th European Conference on Genetic Programming, Lecture Notes in Computer Science*, Springer-Verlag, vol. 2610, pp. 11-59, 2003.

[24] C. C. Bojarczuk, H. S. Lopes, A. A. Freitas, E. L. Michalkiewicz. A Constrained-syntax Genetic Programming System for Discovering Classification Rules: Application to Medical Data Sets. *Artificial Intelligence in Medicine*, vol. 30, no. 1, pp. 27–48, 2004.

[25] P. Angeline. Genetic Programming and Emergent Intelligence. *Advances in Genetic Programming*, vol. 1, pp. 75–98, 1994.

[26] T. Soule. Code Growth in Genetic Programming, Ph. D. dissertation, College of Graduate Studies, University of Idaho, Moscow, Idaho, USA, 1998.

[27] A. L. Garcia-Almanza, E, P. K. Tsang. Simplifying Decision Trees Learned by Genetic Algorithms. In *Proceedings of Congress on Evolutionary Computation*, San Francisco, California, USA, pp. 7906–7912, 2006.

[28] T. Fawcett. Roc Graphs: Notes and Practical Considerations for Researchers, Technical Report HPL-2003-4, HP Laboratories, CA, 2003.

[29] F. J. Provost, T. Fawcett. Robust Classification for Imprecise Environments. *Machine Learning*, vol. 42, no. 3, pp. 203–231, 2001.

[30] C. C. Bojarczuk, H. S. Lopes, A. A. Freitas. Discovering Comprehensible Classification Rules by Using Genetic Programming: A Case Study in a Medical Domain. In *Proceedings of the Genetic and Evolutionary Computation Conference*, Morgan Kaufmann, Orlando, Florida, USA, vol. 2, pp. 953–958, 1999.

[31] J. J. Huang, G. H. Tzeng, C. S. Ong. Two-stage Genetic Programming (2SGP) for the Credit Scoring Model. *Applied Mathematics and Computation*, vol. 174, no. 2, pp. 1039–1053, 2006.

[32] C. Yin, S. Tian, H. Huang, J. He. Applying Genetic Programming to Evolve Learned Rules for Network Anomaly Detection, In *proceedings of the 1st International Conference on Advances in Natural Computation, Lecture Notes in Computer Science*, Springer, Changsha, China, vol. 3612, pp. 323–331, 2005.

Alma Lilia Garcia-Almanza held a first degree in actuary from the Universidad Nacional Autonoma de Mexico, in 1995, and the M. Sc. in computer science from the University of Essex in 2003. She is currently a Ph. D. candidate at the University of Essex.

Her research interests include time series forecasting using genetic programming by means of the analysis of decision trees.

Edward P. K. Tsang is a professor in computer science at University of Essex. He is also the deputy director of Centre for Computational Finance and Economic Agents (CCFEA) and a founding member of the Centre for Computational Intelligence at University of Essex.

He has broad interest in business application of artificial intelligence, including computational finance, computational economics, and constraint satisfaction. Main techniques used included heuristic search, optimization, and evolutionary computation.