ELSEVIER

Expert Systems with Applications 30 (2006) 449–461

Expert Systems with Applications

www.elsevier.com/locate/eswa

Bankruptcy prediction with neural logic networks by means of grammar-guided genetic programming

Athanasios Tsakonas [a,*], George Dounias [b,1], Michael Doumpos [c,2], Constantin Zopounidis [d,3]

[a] Department of Production and Management Engineering, Demokritus University of Thrace, 12 Vas. Sofias St., Xanthi, Greece
[b] Department of Financial and Management Engineering, Business School, University of the Aegean, 31 Fostini Street, 82100 Chios, Greece
[c] Department of Production Engineering and Management, Technical University of Crete, University Campus, 73100 Chania, Greece
[d] Financial Engineering Laboratory, Department of Production Engineering and Management, Technical University of Crete, University Campus, 73100 Chania, Greece

Abstract

The paper demonstrates the efficient use of hybrid intelligent systems for solving the classification problem of bankruptcy. The aim of the study is to obtain classification schemes able to predict business failure. Previous attempts to form efficient classifiers for the same problem using intelligent or statistical techniques are discussed throughout the paper. The application of neural logic networks by means of genetic programming is proposed. This is an advantageous approach enabling the interpretation of the network structure through set of expert rules, which is a desirable feature for field experts. These evolutionary neural logic networks are consisted of an innovative hybrid intelligent methodology, by which evolutionary programming techniques are used for obtaining the best possible topology of a neural logic network. The genetic programming process is guided using a context-free grammar and indirect encoding of the neural logic networks into the genetic programming individuals. Indicative classification results are presented and discussed in detail in terms of both, classification accuracy and solution interpretability.

Keywords: Bankruptcy; Neural logic networks; Grammar-Guided genetic programming; Cellular encoding.

1. Introduction

1.1. The problem of bankruptcy prediction

For more than 30 years, researchers from all over the world, work on the problem of business failure prediction. The problem of timely and correctly predicting bankruptcy, is of great importance for financial institutions. Modeling approaches perform either 'blindly' on a set of data, or with the aid, contribution and guidance of field experts, and vary from classical cross-sectional statistical methods (Balcaen & Ooghe, 2004) to innovative intelligent approaches based on algorithmic data analysis for decision model building (Parag & Pendharkar, 2004). Data are usually based on annual financial statement information (Grice & Ingram, 2001), sometimes used in a straightforward manner as continuous variables (Zhang, Hu, Patuwo & Indro, 1999) and some others represented as discretized and thus, qualitative information (Dimitras, Slowinski, Susmaga & Zopounidis, 1999). However, in a few approaches qualitative variables are also collected and taken into account (Kim & Han, 2003). Comparisons of results are difficult indeed, as the sample size and the number of representative variables vary in literature.

1.2. Background and literature review

Several previous studies on business failure prediction have appeared in literature during the last three decades. The first methodological attempt to model the problem of bankruptcy prediction took place by (Altman, 1968). Below we summarize the main findings from various studies, as well as the sample size, Type of decision variables used and also the accuracy obtained in different experimentations.

In their study, (Balcaen & Ooghe, 2004) discriminate four general Types of classical statistical methods applied in corporate failure prediction, (a) univariate analysis, (b) risk index models, (c) multivariate discriminant analysis, (d)

* Corresponding author. Tel.: + 30 2108222032; fax: +30 2271035499.
E-mail addresses: tsakonas@pme.duth.gr (A. Tsakonas), g.dounias@aegean.gr (G. Dounias), mdoumpos@dpem.tuc.gr (M. Doumpos), kostas@dpem.tuc.gr (C. Zopounidis).
[1] Tel.: +30 2271035454; fax: +30 2271035499.
[2] Tel.: +30 2821037318; fax: +30 2821069410.
[3] Tel.: +30 2821037236; fax: +30 2821069410.

0957-4174/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2005.10.009

conditional probability models. The authors despite the extended use of the classical statistical techniques in literature, find many difficulties in performance due to (a) data anomalies, (b) inappropriate sample selection, (c) matters related to non-stationarity and instability of the data, (d) unreasoned faith and trust on the truth reflected within the financial statements of the firms under consideration, (e) inappropriate selection of independent variables and (f) wrong consideration of the influence of time in the modeling. The work contains a detailed literature review on the bankruptcy problem.

(Parag & Pendharkar, 2004) propose a threshold-varying artificial neural network for binary classification. They compare results to other competitive intelligent techniques such as inductive machine learning and genetic algorithm based neural networks. They use a large set of 1200 cases, but a limited number of attributes (three) and a large number of observations over time (100). Results are mixed for the compared approaches, varying between 71 and 86% for the training set and around 50% for the holdout sample, perhaps due to data difficulties.

(Min & Lee, in press), use machine learning for business failure prediction. Specifically, they propose a methodology based on support vector machines (SVM) and they compare their approach to multiple discriminant analysis, logistic regression analysis and neural networks. They use a large data set of 1888 cases splitting it to training, validation and holdout subsets. The authors also use stepwise logistic regression as a feature selection technique, which gives 11 attributes (i.e. financial ratios) to be used for further modeling of bankruptcy. Support vector machines outperform the other techniques and their performance lies between 71 and 83% for the holdout dataset, depending on the SVM's kernel function selected.

An interesting work is published by (Grice & Ingram, 2001), who explore the generalizability of Altman's *Z*-score model in modern times. They claim that the model is designed for old-style parameters and firm characteristics, so it is not so useful for bankruptcy prediction of nowadays' firms, but it can be still useful for predicting financial stress conditions.

(Philosophov & Philosophov, 2002) attempt to identify both the bankruptcy prediction itself and the time of occurrence, making use of four factors which characterize the quality and quantity of corporate debt and the ability to pay the debt. They apply multivariate analysis based on statistical decision theory, using a medium sample of Russian cases attempting different time predictions varying from 1 to 5 years.

In Park & Han (2002) a case-based reasoning methodology is proposed for corporate failure prediction, namely the analytic hierarchy process weighted K-NN algorithm. The authors use 2144 Korean industry cases, 50% of which represent firms that went bankrupt during 1995–1998. They perform feature reduction using stepwise and *t*-test methods, and they finally cope with a total of 13 attributes. The authors divide the data into training, testing and validation subsets. Results vary from 67 to 83%.

(Cielen, Peeters & Vanhoof, 2004), suggest the combined use of linear programming and inductive machine learning. The authors take into account 367 cases from the Belgian industry and use 11 attributes, selected according to previous successful approaches appeared in literature. They compare their approach to simple linear programming and simple rule induction techniques and they involve 'cost-concepts' in their approach. They report improvement of the overall accuracy when the hybrid method called data envelopment, is used (accuracy varies between 74 and 88% in cross-validation tests).

In Laitinen & Laitinen (2000), a combined use of logistic regression and the Taylor's series expansion is proposed. The methodology is applied in a sample of 400 US companies 50% of which went bankrupt between 1985 and 1993. The authors propose the use of 3 financial ratios and they construct from them in total nine explanatory variables for their logistic model. The accuracies for different statistical models built on the basis of the proposed methodology vary from 67 to 80% when the predictions are made for 1 year prior to failure, but they are reduced to 50–70% when predictions for 2 or 3 years prior to bankruptcy are made.

(Zhang et al., 1999) use neural networks for modeling bankruptcy prediction and they illustrate links to traditional Bayesian classification theory. They use a sample of 220 cases and they initially consider five variables proposed by (Altman, 1968) enriching them later by a few additional ones. Overall classification in validation phase ranges from 77 to 84% proving superior to the logistic model.

(West, Dellana & Qian, in press) suggest the use of three multiplayer perceptron neural network ensemble strategies, namely bagging, boosting and cross validation, for forming financial decision applications. The proposed strategies prove superior to simple neural network approaches, when applied to a set of 329 cases described by the 5 standard financial rations proposed by (Altman, 1968).

(Gorzalczany & Piasta, 1999) compare the effectiveness of a neuro-fuzzy intelligent approach, versus rough sets, for a small sample of 66 cases described by 5 variables. Results are also compared to standard inductive machine learning algorithms.

(Pompe & Bilderbeek, in press) emphasize in their work, the power of specific financial ratios, for the prediction of bankruptcy in small and medium-sized industrial firms. They use 45 ratios to describe the data consisting of various sets of hundreds of cases in different time intervals from the bankruptcy event. Multiple discriminant analysis is applied producing accuracies ranging from 72 to 79%, similar to the performance of neural network approaches as stated by the authors.

(Shin & Lee, 2002) apply a genetic algorithm for extracting meaningful rules for bankruptcy prediction. In their study, they also refer to numerous other artificial applications in bankruptcy. They use nine financial ratios to describe each of their 528 manufacturing cases, but the genetic algorithm based model for prediction uses finally only five of them. Accuracy ranges between 76 and 85% for this promising technique.

(Kim & Han, 2003) also use genetic algorithm based data mining for discovering bankruptcy decision rules from experts' qualitative decisions. The authors use 772 Korean cases and define six qualitative factors to describe the cases.

Comparisons of the rule based outcome to similar ones derived from inductive learning and neural networks are made, showing larger coverage of the data.

(Shin, Lee & Kim, 2005) also use support vector machines for modeling business failure prediction on a sample of Korean firms. The authors use 10 financial ratios to extract the decisions and they find SVMs to be superior compared to neural networks. The work also contains a detailed literature review on the bankruptcy problem.

A hybrid intelligence approach combining genetic programming and rough sets is proposed by (McKee & Lensberg, 2002). The authors use a sample of 291 US firms referring to the period from 1991 to 1997 and they select 11 variables to describe the cases. They conclude that the hybrid model reaches an accuracy of 80% on the validation set, while the simple rough set performs considerably lower on the same data (67%).

(Ahn, Cho & Kim, 2000) work on the combination of rough sets and neural networks for business failure prediction. They also use Korean data referring to the period between 1994 and 1997 and they compare their results to different standard neural network techniques. Accuracy exceeds 80% in some cases.

(Salcedo-Salcedo-Sanz et al., 2005) propose genetic programming for the prediction of possible bankruptcy of the insurance companies. The sample comprises of 72 Spanish insurance firms equally balanced between bankrupt and non-bankrpupt ones, and 21 financial ratios are used to describe the data. Not all the ratios are used by the genetic programming approach to form the decision model, while accuracy is promising. Comparisons are made with rough sets approaches.

Similarly with above, (Lensberg, Eilifsen & McKee, in press) develop a bankruptcy classification model using genetic programming (GP). They use 28 potential variables to describe the data (six of them prove significant) consisting of 422 Norwegian firms for the period between 1993 and 1998. The model proves 81% accurate in the validation set, slightly better than previous GP approaches on US firms, and also superior when compared to the accuracy of traditional logit models on the same data (77%).

Finally, (Calderon & Cheh, 2002) provide a roadmap for future neural networks research in auditing and risk assessment. The work among other financial problems, reviews that of bankruptcy, through 12 recent studies existing in literature.

It can be easily derived that an enormous amount of approaches, datasets and experimentation settings, exist in literature, regarding the problem of business failure prediction. Most predictions have an accuracy ranging from 65 to 85%. During the previous years standard statistical and clustering/-classification techniques were applied, whereas more advanced intelligent techniques for data analysis appear in the last decade. A very careful presentation and analysis of approximately 160 research reports published between 1932 and 1994 is given by (Dimitras, Zanakis & Zopounidis, 1996). An interesting point seems to be that of comprehensibility of the model used to make the prediction. Another open subject is the need for comparisons of alternative methodologies, in identical conditions (datasets, selection of financial ratios, decisions to be taken, time of prediction, etc.). A data repository could be of great assistance for conducting further research and designing a roadmap for what to use in which occasion.

This paper uses a previously published medium size data set, first used by (Dimitras et al., 1999) regarding 118 Greek firms. The detailed data description follows in the next paragraph. Classification accuracy for this data set according to previous attempts varies from 50 to 76% on the testing set, with respect to time before bankruptcy. Nevertheless, the approach by (Dimitras et al., 1999) contains the drawback of requiring the expert's advice for pre-processing the data, during the phase of attribute selection.

The current paper overcomes the above drawback, proving able to:

(a) Obtain high business failure prediction accuracy.
(b) Work automatically without the interaction to human experts, requiring 12 financial rations as the unique input to be processed.
(c) Produce outcomes, potentially comprehensible as a set of decision rules.

Specifically, we aim to produce competitive solutions, both in terms of the achieved classification accuracy and in their ability to be interpreted into human-understandable classification rules. To accomplish this task, we employ a recent computational intelligence advance named the evolutionary neural logic networks model (Tsakonas, Aggelis, Karkazis & Dounias, 2004). The approach combines neural logic networks (Teh, 1995) with genetic programming (Koza, 1992). Neural logic networks are powerful connectionist systems that have already been applied in various domains (Teh, 1995; Quah, Tan, Teh & Sriniivasan, 1995; Quah, Tan, Raman & Sriniivasan, 1996; Sfetsos, 2000). By their definition, neural logic networks can be easily interpreted into a number of expert rules. These networks can be considered as integration between rule-based expert systems and neural networks (Quah et al., 1996). In order to enable the system to produce arbitrarily sized and connected neural logic networks with interpretability, we adapt a BNF-grammar guided (Naur, 1963) genetic programming approach that uses cellular encoding (Gruau, 1996) to describe the neural logic networks. The interpretability of the derived solutions is ensured by our methodology search among candidate network solutions that maintain network weights, which correspond to specific logical operators. Hence, the solution is extracted by adjusting the topology and altering the nodes of the network instead of attempting simply to adjust the weights, facing the danger to destroy the interpretability of the outcome. Our results in this paper demonstrate the ability of the proposed system of evolutionary neural logic networks to explore easily understandable representations and facilitate the knowledge discovery process.

The paper is organized as follows. In Section 2 describes the data used in the analysis. The fundamentals of the 'pieces' of the proposed hybrid intelligent methodology, are given in Section 3, i.e. the concept of the neural logic networks and the specific genetic programming framework adopted in the paper.

The paragraph covers also a brief review in grammar-guided methodologies for genetic programming and the cellular encoding advances for connectionist systems representation within genetic programming individuals. Section 4 contains the design and the description of the implementation principles regarding the proposed hybrid intelligent system. It also contains the description of the problem domain and information about configuration settings. Section 5 presents the results acquired from the current approach, compares the performance to that of other approaches on the same or other similar data sets and discusses comprehensibility and usability issues. Some important concluding remarks are contained in Section 6.

2. Bankruptcy data description

The total set of 118 cases, consists of two unequal subsets, one representing the training set (80 cases) and the other the testing data set (38 cases).

The first (training) data set comprises of 80 Greek firms, belonging to different industries and activated for more than five years in business. The entire data set can be divided into two equal subsets, including 40 firms that went bankrupt during the period 1986–1990 and 40 non-bankrupt firms for the same time period, chosen in such a way that they belonged to the same industry with the bankrupt ones, and had similar characteristics with them, such as number of employees and total assets.

The training sample was collected according to principles similar to those of Altman's model for bankruptcy data (Grice & Ingram, 2001). Most of the bankrupt firms (42%) belonged to the textile sector, and the rest were balanced among various sectors such as food, wear and footwear, wood, paper, publications, plastics, chemicals, minerals, metallurgical and metal industry, transport vehicles, etc.

Similarly, the second (testing) sample comprises of two equal sub-parts of 19 firms each, one including bankrupt firms and the other non-bankrupt ones for the time period 1991–1993. Most of the bankrupt firms (37%) of the testing sample also belong to the textile sector, while the rest were again balanced among various sectors such as food, wear and footwear, wood, plastics, chemicals, minerals, metal industry and transport vehicles.

Financial statements for the bankrupt companies were collected for a period of 1 year prior to failure. Also data from a similar period were collected for the non-bankrupt firms of the sample. Initially, several financial ratios were calculated and then, according to the experts' advice these ratios were limited to 12 attributes denoting bankruptcy indication. The final attribute selection was made (a) according to the experts' subjective opinion on the special characteristics of the specific sample collected, but also (b) was performed in a way that it would represent adequate information about profitability, managerial performance and solvency ratios regarding the cases processed (Dimitras et al., 1999). The 12 financial ratios considered for bankruptcy prediction were:

1. Net income/Gross profit
2. Gross profit/Total assets
3. Net income/Total assets
4. Net income/Net worth
5. Current assets/Current liabilities
6. Quick assets/Current liabilities
7. [Long term debt + Current liabilities]/Total assets
8. Net worth/[Net worth + Long term debt]
9. Net worth/Net fixed assets
10. Inventories/Working capital
11. Current liabilities/Total assets
12. Working capital/Net worth

No discretization process of the financial ratio data took place from the experts, i.e. the continuous form of the data was used in the current approach, as it resulted from the application of each financial ratio calculation over the financial statements.

3. Methodological issues

3.1. Neural logic networks

A neural logic network is a finite directed graph, consisting of a set of input nodes and an output node. In its 3-valued form, the possible value for a node can be one of three ordered pair activation values (1,0) for true, (0,1) for false and (0,0) for don't know. Every synapse (edge) is assigned also an ordered pair weight (x,y) where x and y are real numbers. An example of a neural logic network and its output value (a,b) of node P is shown in Fig. 1.

$$(a,b) = \begin{cases} (1,0) & \text{if } \sum_{j=1}^{k} a_j x_j - \sum_{j=1}^{k} b_j y_j \geq 1 \\ (0,1) & \text{if } \sum_{j=1}^{k} a_j x_j - \sum_{j=1}^{k} b_j y_j \leq -1 \\ (0,0) & \text{otherwise} \end{cases} \quad (1)$$

The rationale behind neural logic networks is to provide a connectionist system 'equipped' with the following properties:

- The truth table of the output of a node corresponds to the truth table of a logical operation.
- Three-valued logic is supported (true, false and don't know).
- Any elementary network that corresponds to a basic logical operation may be combined with others to form larger networks that can perform complex logical decision tasks.

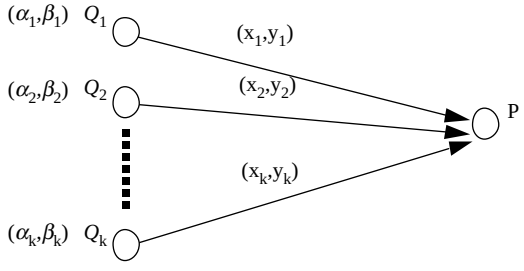

Fig. 1. The general form of a neural logic network and its output value.

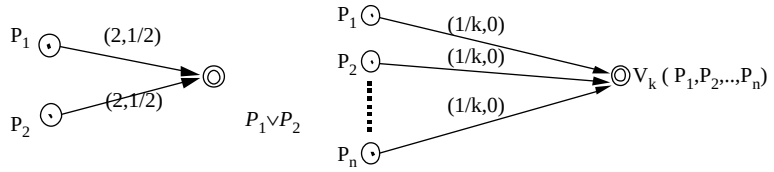

Fig. 2. Example logical operations in neural logic networks.

- Any such neural logic network should be interpretable into logical rules by simply interpreting its architecture and nodes.

In the example shown in Fig. 1, we present the standard activation function for a neural logic node. As it can be seen, the output of such a node belongs to the set {(1,0), (0,1), (0,0)}. By using specific weights, different logical operations can be applied to the input nodes. Then, the result to the output node will be the same as defined in the truth table of the corresponding logical operation.

Different sets of weights enable the representation of different logical operations. It is actually possible to map any rule of conventional knowledge into a neural logic network. In Fig. 2, two examples of logical operators and their implementation in neural logic networks is shown.

3.2. Automatic interpretation of the neural logic networks into Prolog expert rules

According to (Teh, 1995), the neural logic networks fulfil all the features that are required for the unification of the symbolic and neural processing. In the following example, we present an enhancement of the PROLOG programming language, using neural logic networks, aiming at a more powerful programming environment, the so-called 'Neural Prolog' (Teh, 1995). This procedure is illustrated here by the example network of Fig. 3. The interpretation includes the following steps:

Step 1:We enhance the facts of PROLOG, by allowing to simple predicates to get three values: (1,0), (0,1) and (0,0). Hence,

Father(X,Y)=(1,0) means that it is true that X is Father of Y,

Father(X,Y)=(0,1) means that it is false that X is Father of Y,

Father(X,Y)=(0,0) means that it is still unknown if X is Father of Y

If in the same program appear both predicates Father(X,Y)=(1,0) and Father(X,Y)=(0,1), then the program is considered inconsistent. On the opposite, if the predicates Father(X,Y)=(1,0) and Father(X,Y)=(0,0) appear, then the program is consistent and the Father(X,Y)=(0,0) will be deleted. This generalization can be expanded into fuzzy neural logic networks as well.

Step 2: We may create rules into the programming language PROLOG directly by every neural logic network. For example, the neural logic network shown in Fig. 3, which corresponds to a priority rule, produces a number of rules in PROLOG (If-Then rules), which are presented in Fig. 4.

From this example it is shown that every neural logic network may be interpreted into a set of If-Then rules of the PROLOG programming language, which is used mainly in artificial intelligence expert systems and it is easily understandable by humans. The main drawback of the neural logic network when used within the computational intelligence framework was that the application of any learning scheme was destroying the solution interpretability (Teh, 1995). These problems have been overcome using the genetic programming approach that is also applied in this paper (Tsakonas et al., 2004).

3.3. Genetic programming

Genetic programming (GP) is a search methodology belonging to the family of evolutionary computation (EC). Nowadays these algorithms have been applied in a wide range of real-world problems. Genetic programming in its canonical form enables the automatic generation of mathematical expressions or, so-called, 'programs'. According to the most common implementations, a population of candidate solutions is maintained, and after the completion of a 'generation', the population is expected to be better fit a given problem. In standard 'generational' genetic algorithms (GA), a generation consists of the application of genetic operators for every individual; this results into a new population. In contrast to generational GAs, the term 'generation' in steady-state genetic programming (where only one population is maintained) is used to roughly describe a number of algorithm iterations equal to the number of the population. Usual termination criteria appear to be

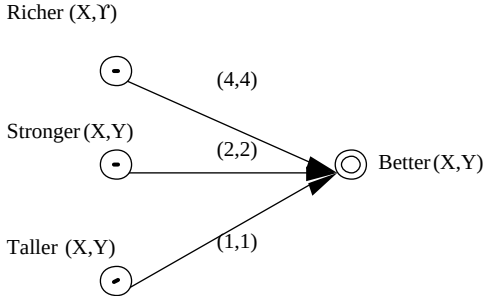

Fig. 3. A neural logic network example, corresponding to a priority rule.

If *richer* (X,Y)=(1,0) then *better* (X,Y)=(1,0)
If *richer* (X,Y)=(0,1) then *better* (X,Y)=(0,1)
Suppose *richer* (X,Y)=(0,0)
    If *stronger* (X,Y)=(1,0) then *better* (X,Y)=(1,0)
    If *stronger* (X,Y)=(0,1) then *better* (X,Y)=(0,1)
    Suppose *richer* (X,Y)=(0,0) and *stronger* (X,Y)=(0,0)
        If *taller* (X,Y)=(1,0) then *better* (X,Y)=(1,0)
        If *taller* (X,Y)=(0,1) then *better* (X,Y)=(0,1)
            Suppose *richer* (X,Y)=(0,0) and *stronger* (X,Y)=(0,0)
            and *taller* (X,Y)=(0,0) then *better* (X,Y)=(0,0)

Fig. 4. Rules in PROLOG that derive by the network of Fig. 3.

the accomplishment of a number of generations, the achievement of a desired classification error, etc. Genetic programming uses tree-like individuals that can represent mathematical expressions, making the application of GP in symbolic regression problems valuable.

The main advantage of genetic programming over genetic algorithms, is their ability to construct functional trees of varying length. This property enables the search for complex solutions that are usually in the form of a mathematical formula—an approach that is commonly known as symbolic regression. Later paradigms have extended this concept to calculate any boolean or programming expression. Thus, complex intelligent structures, such as fuzzy rule-based systems or decision trees have already been used as the desirable target solution in genetic programming approaches (Alba, Cotta & Troya, 1996; Tsakonas & Dounias, 2002a; Tsakonas, Dounias, Axer & von Keyserlingk, 2001; Tsakonas & Dounias, 2002b). The main advantage of this solving procedure is that the feature selection, and the system configuration, occur during the normal run and do not require any human pre-processing.

Although powerful in its definition, the genetic programming procedure might prove greedy in computational and time resources. Therefore, when the syntax form of the desired solution is already known, it is useful to restrain the genetic programming process from searching solutions with different syntax forms (Tsakonas & Dounias, 2002a; Gruau, Whitley & Pyeatt, 1996). The most advantageous method to implement such restrictions among other approaches (Montana, 1995), is to apply syntax constraints to genetic programming trees, usually with the help of a context-free grammar declared in the so called Backus-Naur-Form (BNF) (Naur, 1963; Gruau et al., 1996; Janikow, 1996; Ryan, Collins & O'Neil, 1998). The BNF-grammar consists of terminal nodes and non-terminal nodes and is represented by the set {N, T, P, S}, where N is the set of non-terminals, T is the set of terminals, P is the set of production rules and S is a member of N corresponding to the starting symbol. The construction of the production rules can be the most critical point in the creation of a BNF grammar, since these production rules express the permissible structures of an individual.

The application of massively parallel processing intelligent systems (such as the neural logic networks) is not forthright within the GP-tree framework. In order to explore variable sized solutions, we have applied indirect encoding. The most common one is the cellular encoding (Gruau et al., 1996), in which a genoType can be realised as a descriptive phenoType for the desired solution. More specifically, within such a function set, there are elementary functions that modify the system architecture together with functions that calculate tuning variables. A similar technology, called edge encoding (developed by J. Koza, Bennett, Andre & Keane, 2003) a pioneer in genetic programming (Koza, 1992; Koza et al., 1999), is also used today with human competitive results in a wide area of applications.

4. Design and implementation of the hybrid intelligent system

4.1. Evolutionary neural logic networks

As mentioned above, a hybrid intelligent scheme is proposed for the problem of bankruptcy prediction, combining neural logic networks and genetic programming. The characteristic attribute of neural logic networks is the possibility to interpret any valid and interpretable architecture. For this reason, the cellular encoding is used to represent the candidate solutions into genetic programming trees. One cellular encoding scheme includes two function classes:

- Functions for architecture altering
- Functions for parameter tuning

The functions for parameter tuning have common properties with the usual genetic programming functions, which operate as procedures or program elements. The functions that are used for architecture altering however are not used in standard genetic programming systems. They comprise a function set that alters an embryonic neural logic network, by entering nodes sequentially or in parallel onto an initial (elementary) neural logic network, in order to form the final/desirable architecture. Hence, among the architecture altering functions we may discriminate between (a) functions that enter a node serially, and (b) functions that enter a node in parallel. The problem that has arisen during the prime implementations of cellular encoding concerns the grammar description, which enabled the existence of networks without inputs (Gruau, 1996), a situation that could easily lead into premature population convergence. Thus, in order to be able for a system to include at least one input, we decided to use different functions for the architecture altering on the system inputs, than those used for the architecture altering on internal nodes. Hence, we may further divide the architecture altering functions into two additional sub-classes:

- Functions that are applied on the system inputs
- Functions that are applied on internal nodes

To conclude with, we use a function that enters a node in serial to an input node (S1), a function that enters a node in parallel to an input node (P1), a function that enters a node in serial to an internal node (S2) and, finally, a function that enters a node in parallel to an internal node (P2). In the following Fig. 5, we demonstrate the operation regarding function S1. The application of this function on the B node in Fig. 5(a), results in the construction of the network shown in Fig. 5(b). The grey-colored arrow shows the running cursor, which marks the point from which any further network expansion will occur.

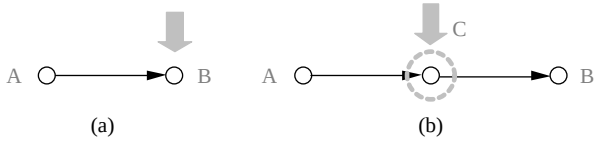

Fig. 5. Application of function S1 on the B node.

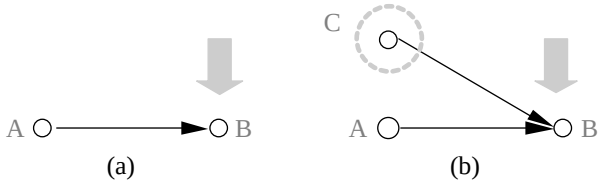

Fig. 6. Application of function P1 on the B node.

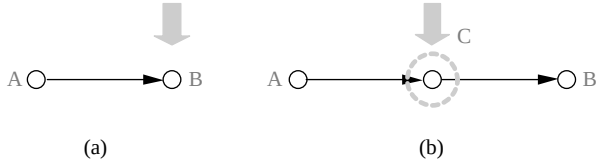

Fig. 7. Application of function S2 on the B node.

Fig. 6 illustrates the operation of P1 function. Application of this function on the B node in Fig. 6(a), results to the network shown in Fig. 6(b). The following Fig. 7 depicts the operation for function S2. The application of this function on the B (internal) node in Fig. 7(a), results to the network shown in Fig. 7(b).

Finally, Fig. 8 shows the operation of P2 function. The application of this function on the B node in Fig. 8(a), results to the network shown in Fig. 8(b).

Table 1 depicts in short the functions that we used in order to describe the evolutionary neural logic networks. It is worth to note that although a genetic programming tree is executed depth-first, the execution of the resulted network is accomplished breadth-first, in order to enable the parallel operation which is a characteristic of a network. In order to implement such a procedure we make use of a parameter list. Additionally, we create a running node cursor, in which all the logical operators are applied.

The system grammar is presented in Table 2. Initial symbol (root) of a tree can be a node of a Type <PROG>.

The logical functions that construct in this work the operator set for the neural logic networks expressed in our system are shown in Table 3.

For a number of logical operations performed onto a neural logic network's node, a single-step procedure is sufficient. However, functions such as the 'exclusive OR' (XOR) or the 'exactly k-true' operator need a two-step procedure for implementation within the neural logic network framework (Teh, 1995).

For example, to implement the 'negative conjunction' (NAND) operator, the following Eq. (2) is adequate.):

$$C = \sum_{i=0}^{p-1} \left(-\frac{f_i}{p-1} + s_i \cdot 2 \right), \quad \forall i : q_i \neq -1 \tag{2}$$

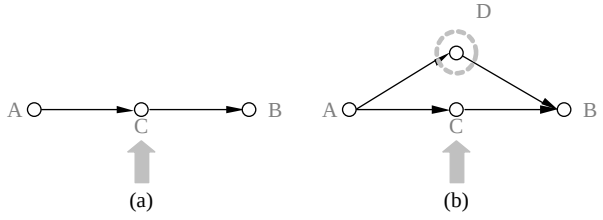

Fig. 8. Application of function P2 on the B node.

Table 1
Functions for the production of neural logic networks within a genetic programming system

Class	Function name	Description
Architecture altering	P1	Enters a node in parallel to an input node
	S1	Enters a node in serial to an input node
	P2	Enters a node in parallel to an internal node
	S2	Enters a node in serial to an internal node
	PROG	Initial function. Creates the embryonic network
Parameter tuning	CNR	Applies logical operator based on the CNRSEL and *K* values
	LNK	Cuts links based on the NUM and CUT values
	IN	Enters an input parameter value into the network
	NUM	Supplementary to the LNK function, selects the link to cut
	CUT	Supplementary to the LNK function, determines whether the link will be cut or not
	CNRSEL	Supplementary to the CNR function, its value determines the operator that will be applied to a node
	K	Supplementary to the CNR, function, its value determines the logical operator's parameter (determined by CNRSEL).

A detailed description for all logical operators can be found in Calderon & Cheh (2002). In order to apply the 'exclusive OR' (XOR) operator however, the following intermediate calculations are used:

$$C1 = \sum_{i=0}^{p-1} \left(\frac{f_i}{2} - \frac{s_i}{2} \right), \quad \forall i : q_i \neq -1 \tag{3}$$

$$C2 = \sum_{i=0}^{p-1} \left(2 \cdot f_i - \frac{s_i}{2} \right), \quad \forall i : q_i \neq -1 \tag{4}$$

$$C = -2 \cdot f_1 + f_2 - s_2 \tag{5}$$

Table 2
Grammar for neural logic networks in BNF notation

<PROG>:=PROG <PLACE1><SYNAPSE>
<PLACE1>:=S1 <PLACE1><SYNAPSE><PLACE2>
| P1 <PLACE1><PLACE1>
| IN
IN: =Data attribute (system input)
<PLACE2>:=S2 <PLACE2><SYNAPSE><PLACE2>
| P2 <PLACE2><SYNAPSE><PLACE2>
| E
E: =∅
<SYNAPSE>:=LNK <NUM><CUT><SYNAPSE>
| CNR <CNRSEL><K>
<NUM>:= NUM
<CUT>:=CUT
<CNRSEL>:=CNRSEL
<K>:=K
NUM: =Integer in [1,256]
CUT: =Integer in [0,1]
CNRSEL: =Integer in [0,10]
K: =Integer in [0,9]

Table 3
Logical operators used within the evolutionary neural logic networks

Function	Steps
Conjunction (AND)	1
disjunction (OR)	1
Priority	1
at least k-true	1
at least k-false	1
Majority influence	1
Majority influence of k	1
2/3 majority	1
Unanimity	1
IF-Then (Kleene' s model)	1
(logical) difference	1
Exclusive OR (XOR)	2
Negative exclusive OR (XNOR), or equivalence (EQV)	2
negative conjunction (NAND)	1
negative disjunction (NOR)	1
exactly k-true	2

where,

$$(f_1,s_1)=(1,0),\ \text{if } C1\geq 1 \qquad (f_2,s_2)=(1,0),\ \text{if } C2\geq 1$$
$$(f_1,s_1)=(0,1),\ \text{if } C1\leq -1 \qquad (f_2,s_2)=(0,1),\ \text{if } C2\leq -1$$
$$(f_1,s_1)=(0,0),\ \text{if } -1<C1<1 \qquad (f_2,s_2)=(0,0),\ \text{if } -1<C2<1$$

Additionally, in order to apply the 'exactly-k-true' operator, the following computations are applied:

$$C1=\sum_{i=0}^{p-1}\left(\frac{f_i}{k}\right) \quad \forall\, i: q_i\neq -1(\text{at least } k-\text{true}) \tag{6}$$

$$C2=\sum_{i=0}^{p-1}\left(-\frac{s_i}{p-1-k}\right) \tag{7}$$

$$\forall i: q_i\neq -1(\text{at least } p-1-k \text{ false})$$

$$C=\sum_{i=0}^{1}(f_i-s_i\cdot 2)(\text{conjunction}-\text{AND}) \tag{8}$$

where,

$$(f_1,s_1)=(1,0),\text{if } C1\geq 1 \qquad (f_2,s_2)=(1,0),\text{if } C2\geq 1$$
$$(f_1,s_1)=(0,1),\text{if } C1\leq -1 \qquad (f_2,s_2)=(0,1),\text{if } C2\leq -1$$
$$(f_1,s_1)=(0,0),\text{if } -1<C1<1 \qquad (f_2,s_2)=(0,0),\text{if } -1<C2<1$$

In all the above equations, f_i denotes the first part of the i-link's pair value, s_i is used for the second part of the i-link's pair value, p is the number of the links ending to the selected node, k is a parameter number (obtained by function K), and q_i denotes the cut value for the i-link (a value of -1 marks the link as deleted). The pair of inputs is taken according to the value of C and the following equation:

$$(x,y)=\begin{cases}(1,0)\ \text{if } C\geq 1\\(0,1)\ \text{if } C\leq -1\\(0,0)\ \text{if } |C|<1\end{cases} \tag{9}$$

Finally, the pair of inputs is encoded in order for the function to return the value. Function CNR actually returns the value that is calculated by the following equation:

$$\text{Res}=2\cdot x+y \tag{10}$$

Obviously, the value Res will be an integer in the interval [0,2], which describes the logic output of the function applied to the node. The system for the fuzzy extension of neural logic networks makes use of the same grammar as the 3-valued model.

However, while in 3-valued neural logic networks the passing values between functions and the list Q are encoded into one integer value, for the fuzzy model this technique is not possible. Hence, we adapt a structure (struct) passing scheme. Also, the CNR function is updated to reflect the sophisticated algorithm which the fuzzy NLNs make use of (Ref. 6). Results for both models are the same if only 3-valued input exists (i.e. only (0,0),(1,0) or (0,1) input data).

4.2. Data preprocessing and system configuration

As stated in the previous paragraph, the fuzzy model for the evolutionary neural logic network allows the processing of any kind of data (i.e. not only boolean logic data). In order to process boolean logic problems, using only the values of true, false and don't know, it is sufficient to apply the simple encoding scheme shown in Table 4.

For the processing of problems that use real-valued data, we actually employ the fuzzy extension of the model, which has the ability to process real-valued data belonging in the range $\{0\}\cup[1,2]$. As stated in the previous paragraph, the fuzzy extension allows the quantification of the 'true' and 'false' content of the value pair. Hence, in Table 5 we present the applied encoding for the fuzzy model.

As it can be observed for the Table 5, the 'true' content and the 'false' content of a value in the fuzzy model in our encoding scheme are complementary. In other words, $\forall(x,y)$, $x+y=1$. In general however, the fuzzy model has the ability to process any value set in which $\forall(x,y)$, $x+y<=1$. For example, a value pair of (0.345,0.441) can be perfectly processed, and its interpretation is False 34.5%, True 44.1% and Don't Know 21.4%. In order to adapt the data in the form needed for the fuzzy model (e.g. in the range $\{0\}\cup[1,2]$) we apply the following procedure.

If the value in the initial data set is unkown, we set the value 0, which corresponds (as shown above) to the value pair (0,0) — 'don't know'.

If the value is any other number (real, known) we apply the midrange–range standardization, a common procedure to normalize the data for input processing in neural networks. This procedure is consisted actually by the calculation of

Table 4
Data usage in the (boolean) neural logic network model

Data value	Value as used in the neural logic network	Interpretation
0	(0,0)	Don't know
1	(0,1)	False
2	(1,0)	True

Table 5
Data usage in the fuzzy neural logic network model

Data value	Value as used in the fuzzy neural logic network	Interpretation
0	(0,0)	Don't know
1	(0,1)	False
2	(1,0)	True
Real ∈ [1, 2] (eg. here: 1. 345)	(0.345,0.655)	False 34.5% True 65.5%

the following values:

$$\text{midrange} = \frac{\max_i X_i + \min_i X_i}{2} \tag{11}$$

$$\text{range} = \max_i X_i - \min_i X_i \tag{12}$$

$$S_i = \frac{X_i - \text{midrange}}{(\text{range}/2)} \tag{13}$$

where S_i is the 'standardized' value and X_i is the current record value. The min X_i and max X_i are the lowest and highest feature value. The values returned belong to the range $[-1,1]$ thus with the simple following calculation, we get the values to be processed by the system:

$$V_i = \frac{S_i + 3}{2} \tag{14}$$

This last computation returns the values V_i in the range [1,2].

Most of the GP parameters of our implementation can be considered as typical for the GP framework (e.g. tournament of 6 individuals, elitist strategy, 100 generations run, etc.). Although the fine-tuning of our algorithm was not the main concern of this paper, we investigated various initialization and run approaches. Without claiming optimality, these GP parameters are presented in Table 6. This setup offered for the presented grammars stable and effective runs throughout experiments. As it can be observed, our setup denotes our preference for relatively high mutation rates, especially the shrink mutation, which in our experiments delayed the effect of code bloat, the latter being typically generated by crossover operations. Although the initialization of the population is random, we used a probability bias towards the GP tree 'leaves', in order to enforce the algorithm to generate individuals of acceptable size.

Table 6
GP system parameters

Parameter	Value
Population:	5,000 individuals
GP implementation:	Steady-state G^3P
Selection:	Tournament with elitist strategy
Tournament size:	7
Crossover rate:	0.65
Overall mutation rate:	0.35
Node mutation rate:	0.4
Shrink mutation rate:	0.6
Killing Anti-Tournament size:	2
Maximum allowed individual size:	1000 nodes
Maximum number of generations:	100

Currently, the most popular procedure in literature to avoid overfitting in the training set is the use of a validation set. This technique consists of the partition of the subset used for training into two parts. The first part is used for the main training of the algorithm, and the second part is used for validation. In this respect, as best solution is selected the one that maintains the lowest classification error in *both* the training and the validation set. More specifically, this procedure in ENLN is applied as follows.

Suppose$F_i^t, F_i^v, \quad i = 1, \ldots, n$, where F_i^t is the fitness value of the best individual in the training set after i generations, F_i^v is the fitness value of the best individual in the validation set after i generations and n is the maximum number of generations, we get the following Eq. (15):

$$V_{\text{best}} \Leftarrow V_i, \quad \text{iff}\left\{\left(F_i^t > F_{\text{best}}^t\right) \wedge \left(F_i^v \geq F_{\text{best}}^v\right)\right\} \tag{15}$$

where V_{best} is the (final) best solution, V_i is the best solution after i generations, F_{best}^t is the fitness value of the (final) best individual in the training set and F_{best}^v is the fitness value of the (final) best individual in the validation set. In fact, aiming to obtain smaller size of the solutions we slightly modify the above Eq. (15) to the following one (16):

$$V_{\text{best}} \Leftarrow V_i,$$

$$\text{iff}\left\{\begin{array}{c}(F_i^t > F_{\text{best}}^t) \wedge (F_i^v \geq F_{\text{best}}^v) \\ \vee \\ (F_i^t = F_{\text{best}}^t) \wedge (F_i^v \geq F_{\text{best}}^v) \wedge (K_i < K_{\text{best}})\end{array}\right\} \tag{16}$$

In the above Eq. (16), K_i is the size (in nodes) of the best solution after i generations, and K_{best} is the size (similarly in nodes) of the final best solution acquired.

5. Results and discussion

We performed 10 runs using the configuration mentioned in previous paragraph. The best of our results are shown in Table 7, together with those found in literature (Dimitras et al., 1999). More specifically, we compare our result with those derived by a rough set approach, a discriminant analysis model and a logit model. We used the same records for training and test as it happens in similar approaches in the existing literature. We divided the 80 training set records into two sets, a training set consisted of 60 records and a validation set with the rest 20 records, used to avoid overfitting. The selection of the validation set was performed randomly (every 4th record was assigned to the validation set). The classification results obtained from the developed neural logic network model are shown in Table 7. For comparison reasons the results of other methods applied in the same data are also reported. The developed model is described in Fig. 9, and depicted in Fig. 10. The solution description shown in Fig. 9 is actually a serial representation of the evolved genetic programming tree, which describes the network shown in Fig. 10. That network can be drawn easily, using a step-by-step procedure as presented in paragraph 4.1. The derived network, in its turn, clearly shows the solution's rule set, since the network weights, in our system, always correspond to logical rules. Hence, a node in

Table 7
Classification accuracy for the neural logic networks

Classification accuracy			
System	Accuracy	Learning sample (%)	Testing sample (%)
Rough sets, minimal set of rules	Bankrupt firms	100	84.2
	Healthy firms	100	57.9
	Total	100	71.1
Rough sets, set of 'strong' rules	Bankrupt firms	97.5	73.7
	Healthy firms	97.5	57.9
	Total	97.5	65.8
Rough sets, set of 'strong', partly discriminating rules	Bankrupt firms	95.0	94.7
	Healthy firms	90.0	57.9
	Total	92.5	76.3
Discriminant function, attributes of reduced information table	Bankrupt firms	92.5	47.4
	Healthy firms	77.5	73.7
	Total	85.0	60.5
Discriminant function, attributes of complete information table	Bankrupt firms	87.5	63.2
	Healthy firms	92.5	68.4
	Total	90.0	65.8
Logit model, attributes of Reduced information table	Bankrupt firms	87.5	63.2
	Healthy firms	80.0	57.9
	Total	83.7	60.5
Logit model, attributes of complete information table	Bankrupt firms	92.5	63.2
	Healthy firms	87.5	57.9
	Total	90.0	60.5
Evolutionary neural logic networks [this paper]	Bankrupt firms	100	100
	Healthy firms	86.7	55.6
	Total	93.3	76.3

our network is actually a rule in the derived rule set. The results for the rough sets system, the discriminant analysis and the logit model are from (Dimitras et al., 1999), where the same data set was used with the exact learning sample and testing sample setup. Since in our experiments, only the training set, which consists of 60 records, is used as learning sample for the genetic programming approach, the related column of Table 7 contains the classification score of this data set.

As it can be seen from Table 7, the evolutionary neural logic networks achieved the highest classification score in total for the testing sample, together with the model for rough sets with partly discrimination rules.

This solution has a very appealing feature: its success on diagnosing the bankruptcy firms achieved a 100% success in both the learning sample and the testing (unknown) data. The perfect prediction that achieves the system for bankruptcy cases minimizes the cost of misclassifications, since the cost for a bank of classifying a bankrupt case as a healthy one (Type I error) is far more serious than the opposite (Type II error: classifying a healthy case as a bankrupt one). Concluding, among the solutions presented in Table 7 this network has:

- The highest classification accuracy (100%) for the bankrupt firms in the testing sample (unknown data).
- The highest classification overall accuracy (76.3%) for the complete testing sample, which is similar to the performance of rough sets (with 'strong', partly discriminating rules).

The developed model uses only only 6 out the 12 features that were available. The selected attributes are the following:

- Net income/Gross profit
- Net income/Total assets
- Net worth/[Net worth + Long term debt]
- Net worth/Net fixed assets
- Current liabilities/Total assets
- Working capital/Net worth

The selection of the above financial ratios is in accordance with numerous other similar studies found in literature.

The interpretation of the acquired solution yields the following set of expert rules (Q1–Q11 are intermediate results; Q is the concluding rule, which corresponds to the most right network node, that is shown in *Picture* 10 just before the output node):

1. Q1 ← at least 3 false (Net Worth/[Net Worth + Long Term Debt])
2. Q2 ← 2/3 majority (Net Worth/Net Fixed Assets)

```
(ENLN (P1 (P1 (P1 (In X11) (In X3)) (P1 (S1 (In X9) (Link 235 3 (Rule 6 2)) E) (P1 (In X1) (P1 (S1
(In X11) (Rule 6 2) (P2 E (Rule 2 2) E)) (P1 (P1 (S1 (In X12) (Rule 11 3) E) (In X12)) (S1 (P1 (S1 (In
X11) (Link 236 5 (Rule 5 5)) E) (S1 (In X9) (Rule 7 3) E)) (Rule 11 4) (S2 (P2 E (Link 214 2 (Rule 6
3)) E) (Rule 8 13) E))))))) (S1 (In X8) (Rule 4 3) E)) (Rule 12 2))
```

Fig. 9. Extracted solution (description) of the neural logic network for bankruptcy prediction (in postfix notation).

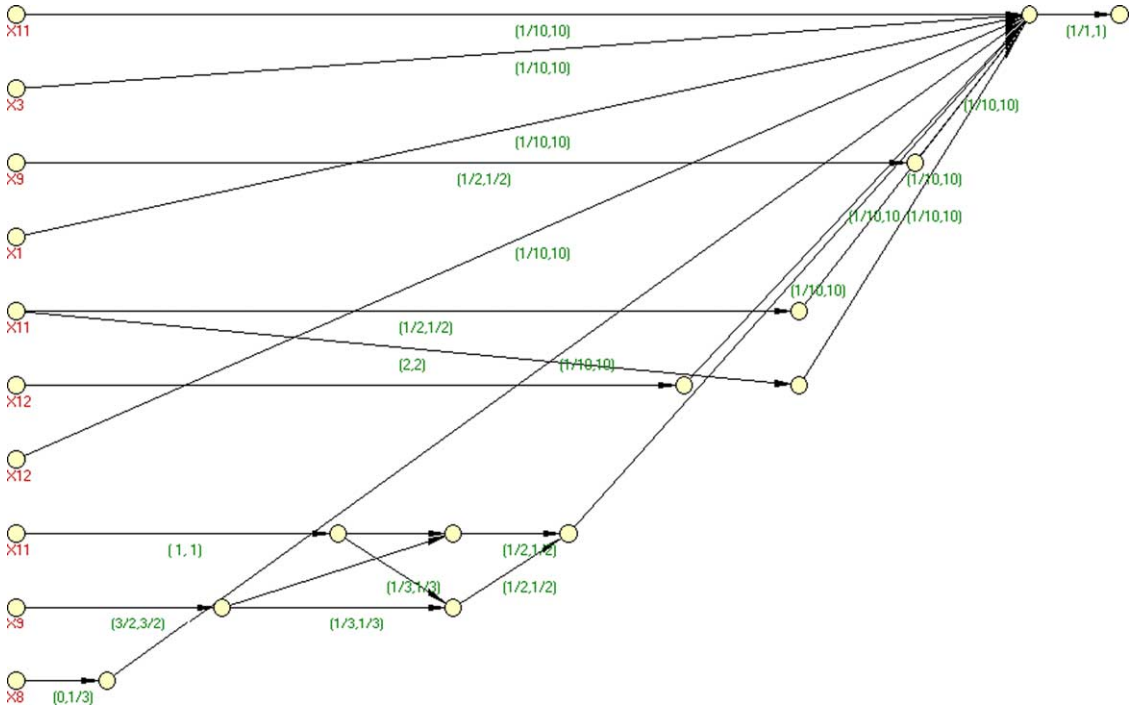

Fig. 10. The neural logic network evolved for the bankruptcy prediction.

3. Q3 ← majority influence (Current Liabilities/Total Assets)
4. Q4 ← exclusive or —XOR (Q2, Q3)
5. Q5 ← majority influence of 3 (Q2,Q3)
6. Q6 ← unanimity (Q4,Q5)
7. Q7 ← majority influence of 3 (Net Worth/Net Fixed Assets)
8. Q8 ← majority influence of 2 (Current Liabilities/Total Assets)
9. Q9 ← priority (Current Liabilities/Total Assets)
10. Q10 ← exclusive or —XOR (Working Capital/Net Worth)
11. Q ← conjunction —AND (Net Income/Gross Profit, Net Income/Total Assets, Current Liabilities/Total Assets, Working Capital/Net Worth, Q1, Q6, Q7, Q8, Q9, Q10)

This solution shows that the variables with the most impact for the system's decision making are the Net Income/Gross Profit, the Net Income/Total Assets, the Current Liabilities/Total Assets, and the Working Capital/Net Worth. Their importance is shown since their values are used in a final conjunction network node (corresponding to a conjunction expert rule). We consider that, although the resulted rule set still needs the fuzzy extension of the neural logic networks for acquiring the decision, this set of comprehensive rules and the promoted features can be potentially used furthermore as generic guide in bankruptcy research. Moreover, due to its high classification rates -especially for the bankrupted firms-, the model itself could be used as a bankruptcy predictor in financial decision-making.

As a concluding remark, we may observe that the evolutionary neural logic networks model proved able to produce a highly competitive solution for the bankruptcy prediction problem addressed in this work. Moreover, in all cases, the extracted networks maintain their interpretability and occasionally, the small size of these networks may discover easy and understandable rules that can potentially discover and establish new expert knowledge in the related field.

6. Conclusions and further work

Within this work an attempt was made to introduce the use of a new hybrid intelligent approach for bankruptcy prediction. The proposed methodology lead to the development of efficient classification schemes formed as simple logic-based decision rules. The proposed hybrid intelligent method is evolutionary neural logic networks by which, evolutionary programming techniques are used for obtaining the best possible topology of a neural logic network. The genetic programming process is guided using a context-free grammar and indirect encoding of the neural logic networks into the genetic programming individuals. The resulting discrimination rules provided satisfactory results in predicting bankruptcy. The proposed methodology outperformed other techniques (either intelligent or, statistical) research attempts in the same dataset in terms of prediction accuracy. In addition, the resulting outcome might prove superior to previously suggested solutions, as it appears to be comprehensible and extremely simple in some cases, and thus attractive for the field experts. Moreover, such a simple model (shaped in the form of meaningful decision rules), could be possibly utilized by field experts in daily practice, as an extension of the existing diagnostic knowledge in the domain. In this sense, evolutionary neural logic networks can be viewed as a hybrid intelligent methodology, not only able to generalize adequately over data, but capable as well of discovering new knowledge for the application domain to which the data correspond.

Further research is in progress in order to extensively explore the solution space and possibly provide more accurate results

regarding the domain of bankruptcy prediction. In bankruptcy prediction models literature, Type I accuracy is given more weight than Type II, hence we consider that using a weighted sum of accuracies may further improve our results. Additional (new) data collection is in progress in order to be used for demonstrating the effectiveness of the proposed approach and the ability to produce generalized knowledge, applicable in different datasets. This will enable the direct comparison with other models often used in bankruptcy prediction (e.g. logit models). Moreover, experimenting using larger or smaller data sets will enable a clearer view of the general system performance. Finally, repeated experimentation related to the tuning of the overall genetic programming parameters (corresponding to a really hard task in terms of computational effort) is in progress, in order to obtain the more efficient possible search procedure of the algorithm.

Evolutionary neural logic network-based methodologies have also been used from the authors, in a variety of application domains with diverse complexity and characteristics, including but not limited to, financial decision support, credit risk management, medical decision making and modeling of fault diagnosis in engineering applications. The approach is always case-sensitive, i.e. has to be designed and tailored properly according to the special characteristics of the application domain.

Nevertheless, a number of open issues exist for the extension of the proposed methodology, and these are stated below:

- Construction of evolutionary recursive high-order neural networks, maintaining the interpretability of the outcomes
- Application of similar methodologies in multi-class problems and multivariate time-series financial forecasting problems
- Research in the direction of recording the opinion of financial experts, regarding the acknowledgement of the proposed mechanism, as an effective method for potential discovery of new (expert) knowledge

References

Ahn, B. S., Cho, S. S., & Kim, C. Y. (2000). The integrated methodology of rough set theory and artificial neural network for business failure prediction. *Expert Systems with Applications*, *18*, 65–74.

Alba, E., Cotta, C., & Troya, J. M. (1996). Evolutionary design of fuzzy logic controllers using strongly-typed GP. In *Proceedings of 1996 IEEE international symposium on intelligent control* (pp. 127–132). New York, NY.

Altman, E. I. (1968). Financial rations, discriminant analysis and the prediction of corporate bankruptcy. *The Journal of Finance*, *23*(4), 589–609.

Balcaen, S., & Ooghe, H. (2004). 35 years of studies on business failure: An overview of the classical statistical methodologies and their related problems. Working paper 248, Department of Accountancy and Corporate Finance, Ghent University, Belgium.

Calderon, T. G., & Cheh, J. J. (2002). A roadmap for future neural networks research in auditing and risk assessment. *Information Systems*, *3*, 203–236.

Cielen, A., Peeters, L., & Vanhoof, K. (2004). Bankruptcy prediction using a data envelopment analysis. *European Journal of Operational Research*, *154*(2), 526–532.

Dimitras, A. I., Slowinski, R., Susmaga, R., & Zopounidis, C. (1999). Business failure prediction using rough sets. *European Journal of Operational Research*, *114*, 263–280.

Dimitras, A. I., Zanakis, S. H., & Zopounidis, C. (1996). A survey of business failures with an emphasis on prediction methods and industrial applications. *European Journal of Operational Research*, *90*, 487–513.

Gorzalczany, M. B., & Piasta, Z. (1999). Neuro-fuzzy approach versus rough-set inspired methodology for intelligent decision support. *Information Sciences*, *120*(1–4), 45–68.

Grice, J. S., & Ingram, R. W. (2001). Tests of the generalizability of Altman's bankruptcy prediction model. *Journal of Business Research*, *54*(1), 53–61.

Gruau, F. (1996). On using syntactic constraints with genetic programming. In P. J. Angeline, & K. E. Jinnear (Eds.), *Advances in genetic programming*. Cambridge, MA: MIT Press.

Gruau, F., Whitley, D., & Pyeatt, L. (1996). A comparison between cellular encoding and direct encoding for genetic neural networks. In J. R. Koza, D. E. Goldberg, D. B. Fogel, & R. L. Riolo, *Genetic programming 1996: Proceedings of the first annual conference* (pp. 81–89). Cambridge, MA: MIT Press.

Janikow, C. Z. (1996). A methodology for processing problem constraints in genetic programming. *Computers and Mathematics with Applications*, *32*(8), 97–113.

Kim, M.-J., & Han, I. (2003). The discovery of experts' decision rules from qualitative bankruptcy data using genetic algorithms. *Expert Systems with Applications*, *25*(4), 637–646.

Koza, J., Bennett, F., & Keane, M. (2003). *Genetic programming IV: Automatic programming and automatic circuit synthesis*. San Francisco, CA: Morgan Kaufmann.

Koza, J. R. (1992). *Genetic programming: On the programming of computers by means of natural selection*. Cambridge, MA: MIT Press.

Koza, J. R., Bennett, F. H., & Keane, M. A. (1999). *Genetic programming III*. San Francisco, CA: Morgan Kaufmann.

Laitinen, E. K., & Laitinen, T. (2000). Bankruptcy prediction: Application of the Taylor's expansion in logistic regression. *International Review of Financial Analysis*, *4*(4), 327–349.

Lensberg, T., Eilifsen, A., & McKee, T. E. (in press). Bankruptcy theory development and classification via genetic programming. *European Journal of Operational Research*, p. 21. visit: www.elsevier.com/locate/dsw.

McKee, T. E., & Lensberg, T. (2002). Genetic programming and rough sets: A hybrid approach to bankruptcy classification. *European Journal of Operational Research*, *138*(2), 436–451.

Min, J. H., & Lee, Y.-C. (in press). Bankruptcy prediction using support vector machine with optimal choice of kernel function parameters. *Expert Systems with Applications*. Uncorrected proof, available online 5 January 2005.

Montana, D. J. (1995). Strongly typed genetic programming. *Evolutionary Computation*, *3*(2).

Naur, P. (1963). Revised report on the algorithmic language ALGOL 60. *Communications of the ACM*, *6*(1), 1–17.

Park, C.-S., & Han, I. (2002). A case-based reasoning with the feature weights derived by analytic hierarchy process for bankruptcy prediction. *Expert Systems with Applications*, *23*(3), 255–264.

Philosophov, L. V., & Philosophov, V. L. (2002). Corporate bankruptcy prognosis: An attempt at a combined prediction of the bankruptcy event and time interval of its occurrence. *International Review of Financial Analysis*, *11*(3), 375–406.

Pompe, P. P. M., & Bilderbeek, J. (in press). The prediction of bankruptcy of small- and medium-sized industrial firms. *Journal of Business Venturing*. Corrected proof. Available online: 2 December 2004.

Quah, T.-S., Tan, C.-L., Raman, K., & Sriniivasan, B. (1996). Towards integrating rule-based expert systems and neural networks. *Decision Support Systems*, *17*, 99–118.

Quah, T.-S., Tan, C.-L., Teh, H.-H., & Sriniivasan, B. (1995). Utilizing a neural logic expert system in currency option trading. *Expert Systems with Applications*, *9*(2), 213–222.

Ryan, C., Collins, J. J., & O'Neil, M. (1998). Grammatical evolution: Evolving programs for an arbitrary language. In W. Banzhaf, R. Poli, M. Schoenauer, & T. C. Fogarty, *Genetic programming. Lecture notes in computer science*. London, UK: Springer.

Salcedo-Sanz, S., Fernandez-Villacanas, J.-L., Segovia-Vargas, M. J., & Bousono-Calzon, C. (2005). Genetic programming for the prediction of insolvency in non-life insurance companies. *Computers and Operations Research*, *32*, 749–765.

Sfetsos, A. (2000). A comparison of various forecasting techniques applied to mean hourly wind speed time series. *Renewable Energy*, *21*, 23–25.

Shin, K.-S., Lee, T. S., & Kim, H.-J. (2005). An application of support vector machines in bankruptcy prediction model. *Expert Systems with Applications*, *28*(1), 127–135.

Shin, K.-S., & Lee, Y.-J. (2002). A genetic algorithm application in bankruptcy prediction modelling. *Expert Systems with Applications*, *23*(3), 321–328.

Teh, H.-H. (1995). *Neural logic networks*. Singapore: World Scientific.

Tsakonas, A., Aggelis, V., Karkazis, I., & Dounias, G. (2004). An evolutionary system for neural logic networks using genetic programming and indirect encoding. *International Journal of Applied Logic*, *2*(3), 349–379.

Tsakonas, A., & Dounias, G. (2002). Hierarchical classification trees using type-constrained genetic programming. In *Proceedings of first the international IEEE symposium in intelligent systems*, Varna.

Tsakonas, A., & Dounias, G. (2002). A scheme for the evolution of feedforward neural networks using BNF-grammar driven genetic programming. In Proceedings of Eunite-02, Algarve.

Tsakonas, A., Dounias, G., Axer, H., & von Keyserlingk D. G. (2001). Data classification using fuzzy rule-based systems represented as genetic programming type-constrained trees. In *Proceedings of the UKCI-01*. Edinbourgh (pp. 162–168).

West, D., Dellana, S., & Qian, J. (2004). Neural network ensemble strategies for financial decision applications. *Computers and Operations Research*. Corrected proof. Available online: 25 May 2004.

Zhang, G., Hu, M. Y., Patuwo, B. E., & Indro, D. C. (1999). Artificial neural networks in bankruptcy prediction: General framework and cross-validation analysis. *European Journal of Operational Research*, *116*(1), 16–32.