Evolutionary Neural Logic Networks in Two Medical Decision Tasks.

Athanasios Tsakonas, and Georgios Dounias
University of the Aegean, Dept. of Business Administration,
8 Michalon Str., 82100 Chios, Greece,
Phone: +30-271-35165, Fax: +30-271-93464
e-mail: tsakonas@stt.aegean.gr, g.dounias@aegean.gr

ABSTRACT: Two real-world problems of the medical domain are addressed in this work using a novel approach belonging to the area of neural-symbolic systems. Specifically, we apply evolutionary techniques for the development of neural logic networks of arbitrary length and topology. The evolutionary algorithm is consisted of grammar guided genetic programming using cellular encoding for the representation of neural logic networks into population individuals. The application area is consisted of the diagnosis of patient postoperative treatment and the diagnosis of the Breast cancer. The extracted solutions maintain their interpretability into simple and comprehensible logical rules. The overall system is shown capable to generate arbitrarily connected and interpretable evolved solutions leading to potential knowledge extraction.

KEYWORDS: Neural logic networks, Genetic programming, Postoperative treatment, Breast cancer

INTRODUCTION

Although computational intelligence (CI), has nowadays substituted traditional artificial intelligence in major applications, for a number of high-level decision tasks common expert systems remain still applicable. The reason can be noticed into the need for symbolic representation of the knowledge into these systems, which is a feature that many CI systems have unremarkable success. In other words, it is considered that symbolic representation can be of significant value in these systems for humans, by making clear the inference process to users. Among CI methodologies, neural networks are powerful connectionist systems that still lack the element of complete and accurate interpretation into human-understandable form of knowledge and remain a black box for experts. To deal with this situation, a number of alternative approaches have been proposed. Neural logic networks [1] belong to this category, and by their definition can be interpreted into a number of *Prolog* rules that consist an expert system. Virtually every logic rule can be represented into these networks and then transformed into Prolog commands. Although this model offers excellent results when used within the AI framework (i.e. building a system in a top-down process), the application of neural logic networks in CI's data mining tasks – considered a bottom-up procedure- has undergone limited success. The reason lies in that proposed systems suffered at least one of the following limitations:

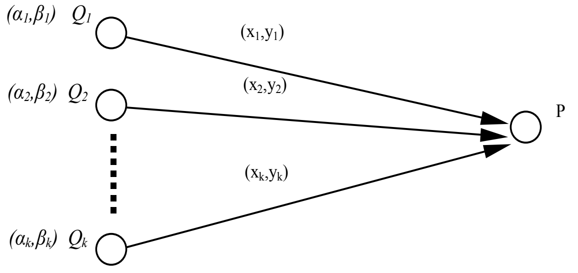
- The extracted neural logic network cannot be interpreted into expert rules [1]-[2].
- The proposed methodology cannot express neural logic networks in their generic graph form [3].
- The user has to select topology and network connection model [1]-[2].

The application of neural logic networks into adaptive tasks seems promising: the extracted model will preserve its interpretability into a number of expert rules and there is not needed any knowledge-acquiring step. Moreover, a solution obtained this way, leads to potential knowledge extraction. Very recently, a new system, namely the evolutionary neural logic networks (ENLN), has been proposed [4] that fulfils those requirements. The new approach uses grammar-guided genetic programming to produce neural logic networks. The evolved solutions can be arbitrarily large and connected networks, since an indirect encoding is adopted. Also, neural logic networks produced by this methodology can always be interpreted into human-understandable expert rules, thus leading to potential knowledge extraction. In this work, we present in brief the methodology and then we apply and test its effectiveness into two real-world problems from the medical domain. In the first task, we examine the system for the diagnosis of Diabetes, and in the second application, we use the system for the diagnosis of the course of Hepatitis patients. The paper is organized as follows. Next section describes the theoretical background, presenting the neural logic networks concept and the grammar guided genetic programming. Following this section, we deal with the design and the implementation of the ENLN system. Next, the results and a following discussion are presented. The paper ends with our conclusion and a description of future work.

BACKGROUND

Neural logic networks

The neural logic network is a finite directed graph. It is usually consisted by 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 neural logic and its output value *(a,b)* of node P is shown in *Fig. 1*. 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. Neural logic networks can be expanded into fuzzy neural logic networks, enabling this way the handling of real valued attributes [1].



(α_1,β_1) Q_1
(x_1,y_1)
(α_2,β_2) Q_2
(x_2,y_2)
(x_k,y_k)
(α_k,β_k) Q_k
P

$$(a,b)=\begin{cases}(1,0) & if\sum_{j=1}^{k}a_jx_j-\sum_{j=1}^{k}b_jy_j\geq 1\\ (0,1) & if\sum_{j=1}^{k}a_jx_j-\sum_{j=1}^{k}b_jy_j\leq -1\\ (0,0) & otherwise\end{cases}$$

Fig. 1. Example neural logic network and its output.

In *Fig. 2*, an example of two logical operators and their implementation in neural logic networks is shown. These operations are part of the logical function set that is used to compose the solutions within the framework of this work.

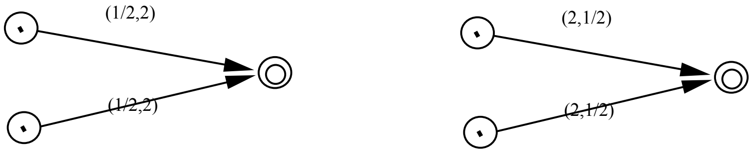
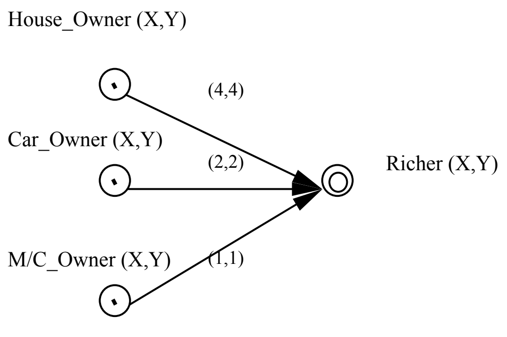


Fig. 2. Example logical operations for operating conjunction and disjunction in neural logic networks.

In order to illustrate the interpretability of neural logic networks into expert rules, let us consider the following simple neural logic network, consisting of the priority rule:

Richer (X,Y) $\Leftarrow$ **priority** (House_Owner, Car_Owner, M/C_Owner)



If *House_Owner*(X,Y)=(1,0) then *Richer*(X,Y)=(1,0)
If *House_Owner* (X,Y)=(0,1) then *Richer* (X,Y)=(0,1)
Supposing *House_Owner* (X,Y)=(0,0)
    If *Car_Owner*(X,Y)=(1,0) then *Richer* (X,Y)=(1,0)
    If *Car_Owner* (X,Y)=(0,1) then *Richer* (X,Y)=(0,1)
    Supposing *House_Owner* (X,Y)=(0,0) and *Car_Owner* (X,Y)=(0,0)
        If *M/C_Owner*(X,Y)=(1,0) then *Richer* (X,Y)=(1,0)
        If *M/C_Owner* (X,Y)=(0,1) then *Richer* (X,Y)=(0,1)
Supposing *House_Owner* (X,Y)=(0,0) and *Car_Owner* (X,Y)=(0,0)
and *M/C_Owner* (X,Y)=(0,0) then *Richer* (X,Y)=(0,0)

Fig. 3. Example neural logic network, corresponding to a priority rule and Prolog rules equivalent to this neural logic network.

Fig.3 depicts the neural logic network corresponding to the above rule. The interpretation of the network into *Prolog* rules is straightforward and the resulted Prolog commands are shown. Even though powerful in their definition, neural logic networks are not widely applied. The main reason can be located in the fact that for the known training methodologies [1]-[2], the refinement of the edge weights reduces significantly the interpretability of these networks to expert rules, thus depriving these networks from their valuable feature. Some steps for the preservation of the interpretability have been performed by [3], without however the ability to express arbitrarily large and connected neural logic networks. For instance, a neural logic network, which performs the important logical operation of XOR, cannot be represented using the direct encoding of [3].

Grammar Guided Genetic Programming

The ability to construct functional trees of variable length is a major advantage of genetic programming over genetic algorithms. This property enables the search for very complex solutions that are usually in the form of a mathematical formula - an approach that is commonly known as symbolic regression. Later paradigms extended this concept to calculate any boolean or programming expression. Consequently, 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 [5]-[8]. The main qualification of this solving procedure is that the feature selection, and the system configuration, derive in the searching process and do not require any human involvement. Moreover, genetic programming, by inheriting the genetic algorithms' stochastic search properties, does not use local search -rather uses the hyper plane search-, and so avoids driving the solution to any local minimum. The potential gain of an automated feature selection and system configuration is obvious; no prior knowledge is required and, furthermore, not any human expertise is needed to construct an intelligent system. Nevertheless, the task of implementing complex intelligent structures into genetic programming functional sets in not rather straightforward. The function set that composes an intelligent system retains a specific hierarchy that must be traced in the GP tree permissible structures. This writing offers two advantages. First, the search process avoids candidate solutions that are meaningless or, at least, obscure. Second, the search space is reduced significantly among only valid solutions. Thus, a genotype - a point in the search space- corresponds always to a phenotype - a point in the solution space. This approach -known as legal search space handling method [9]- is applied in this work using context-free grammars. As we will discuss in the next paragraph, the implementation of constraints using a grammar can be the most natural way to express a family of allowable architectures. While each intelligent system -such as a neural logic network- has a functional equivalent -by means of being composed by smaller, elementary functions-, what defines and distinguishes this system is its grammar.

Context-free grammars

The genetic programming procedure may be proved greedy in computational and time resources. Consequently, when the syntax form of the desired solution is already known, it is useful to restrain the genetic programming from searching solutions with different syntax forms [10]-[11]. The most advantageous method to implement such restrictions among other approaches [12], is to apply syntax constraints to genetic programming trees, usually with the help of a context-free grammar declared in the Backus-Naur-Form (BNF) [13]. 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 use of the terms terminal and non-terminal in a BNF-grammar, does not correspond to what is usually referred in genetic programming as terminal and function. Rather, a function -a non-terminal node in terms of the GP tree architecture- is expressed as terminal in a BNF grammar.

Cellular Encoding

Although mapping decision trees or fuzzy rule-based systems to specific grammars can be relatively easy to implement, the execution of massively parallel processing intelligent systems -such as the neural logic networks- is not forthright. In order to explore variable sized solutions, we applied indirect encoding. The most common one is the cellular encoding [15], in which a genotype can be realized 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. Current implementations include encoding for feed forward and Kohonen neural networks [16], [17] and fuzzy Petri-nets [18], [17]. In his original work, Gruau also used a context-free grammar - a BNF grammar- to encode indirectly the neural networks. On the other hand, in [18] a logic grammar - a context-sensitive one- is adapted to encode fuzzy Petri-nets. In our work, we show that as long as the depth-first execution of the program nodes of a GP tree is ensured -which is the default-, a context-free grammar such as a BNF grammar is adequate for expressing neural networks. Gruau's original work has been facing some skepticism [19] on the ability to express arbitrarily connected networks. Later developments [10] seem to offer less restrictive grammar, though the cut function in those implementations still maintained bounded effect. A similar technology, called edge encoding, developed by [20] is also today used with human competitive results in a wide area of applications.

DESIGN AND IMPLEMENTATION

Data Pre-processing and Genetic Programming Setup

It is accepted that the genetic programming procedure may suffer size problems during initialisation [21]. Although the fine-tuning of our algorithm was not the main concern of this paper, we investigated various initialisation approaches. Without claiming optimality, the GP parameters are presented in *Table II*. This setup, together with function selection probability optimisation, offered for the presented grammar stable and effective runs throughout experiments. The optimisation of function selection probabilities is consisted of giving more selection probability to *GPTerminals* rather than *GPFunctions*. Although the initialisation of the population is random, using this probability bias the algorithm is forced to generate individuals of acceptable size. This optimisation was decided after experimentation, since it is not possible to obtain a general principle regarding the most proper probability values for every case. As it can be observed in *Table II*, the setup denotes our preference for significantly high mutation rates, especially shrink mutation [22] that slows down the code bloat caused by crossover operations.

Table II. GP parameters for ENLN

Parameter	Value
Population:	2,000 individuals
GP implementation:	Steady-state Grammar Guided GP
Selection:	Tournament with elitist strategy
Tournament size:	6
Crossover Rate:	0.35
Overall Mutation Rate:	0.65
Node Mutation Rate:	0.4
Shrink Mutation Rate:	0.6
Killing Anti-Tournament size:	2
Maximum allowed individual size:	650 nodes
Maximum number of generations:	100

System Grammar and Operating Functions

The proposed system grammar is shown in *Fig 4*. Initial symbol (root) of a genetic programming tree can be a node of type `<PROG>`. The function set is as follows:

- **Function `PROG`:** The function `PROG` creates the embryonic network that is used later by the functions `S1`, `S2`, `P1` and `P2` to be expanded. An alternative name for this function, which is used throughout this paper, is the term “`CNLN`”.
- **Function `S1`:** The function `S1` enters a node in serial to the node that is applied, and is applied to input nodes.
- **Function `P1`:**The function `P1` enters a node in parallel to the node that is applied, and is also applied to input nodes..
- **Function `S2`:** The function `S2` enters a node in serial to the node that is applied, and is used for hidden layer nodes.
- **Function `P2`:** The function `P2` enters a node in parallel to the node that is applied, and is also used for hidden layer nodes. This mechanism, consisting of two different sets of expanding functions (`P1` and `S1` vs. `P2` and `S2`), is used to ensure that population individuals will include at least one input node.

- **Function `IN`:** The operation of function `IN` is to assign a variable to the input node that it is applied.

```
<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]
```

Fig. 4. Context free grammar for the production of neural logic networks within genetic programming framework.

- **Function `E`:** The operation of function `E` is to mark the end of the expansion of the network.
- **Function `LNK`:** This function provides the framework for the application of `cut` function. It actually enables the non-full connectivity of the network, a feature that offers larger solution search space.
- **Function `CNR`:** This function performs the node inference. Based on the first parameter, the corresponding calculation is performed. The second parameter assists the calculation for the *at-least-k* and *majority-of-k* operators. Possible computations are shown in *Table III*. An alternative name for this function, which is used throughout this paper, is the term "`Rule`". In order to be able to process values other than *true*, *false* and *don't know*, we applied the *fuzzy propagation algorithm* [1], which allows us to process any real valued variables (using proper normalization).
- **Function `NUM`:** The function `NUM` returns an integer in the interval [1,256] to be used by the calling `LNK` function.
- **Function `CUT`:** The function `CUT` returns an integer in the interval [0,1] to be used by the calling `LNK` function. If the returned value is 1, then the link will be ignored in the calculations (considered "cut").
- **Function `CNRSEL`:** The function `CNRSEL` returns an integer in the interval [0,8] to be used by the calling `CNR` function as its first parameter.
- **Function `K`:** The function `K` returns an integer in the interval [1,256] to be used by the calling `CNR` function if the returned value of the `CNRSEL` is 3,4 or 6 (corresponding to the calculation of the *at least k-true*, *at least k-false* and *majority of k* functions).

Having discussed the system design, in the following session we shall apply the methodology in two medical domains.

Table III. Operations for Function CNR

Parameter	Calculation
0	Conjunction
1	Disjunction
2	Priority
3	At least k-true
4	At least k-false
5	Majority influence
6	Majority influence of k
7	2/3 Majority
8	Unanimity
9	If-Then operation, Kleene's model
10	Difference

RESULTS AND DISCUSSION

Postoperative Treatment

This problem is concerned with the decision whether a patient who is in a postoperative phase, should stay in the hospital or sent home. The database contains 64 patient that must stay in hospital, 2 patients that must be put in intensive care and 24 patients that must be sent home. [25]. In order to develop a classification system we consider as decision task whether a patient should be sent home or stay in the hospital in general. By using this assumption, we include the extremely small class of 'intensive care' patients (2 records) to those of 'usual care in hospital' patients. Thus, the class of 'stay in hospital' patients includes 66 records, and the class of patients sent home the rest 24 [26]. In order to avoid overfitting during the training phase, we applied the minimum description length principle. According to the literature, the target is to develop a system with high accuracy and potential knowledge extraction. *Table IV* presents the domain characteristics, and *Table V* shows the input features in detail.

Table IV. Domain description for the Postoperative treatment problem.

Domain	Patient Postoperative Treatment
Data Base	Post-operative
Input Features	8
Continuous	8
Discrete	0
Binary	0
Anti-Overfitting Method	Minimum Description Length
Total Records	88
Training Set Records	66
Validation Set Records	0
Test Set Records	22
Missing Data	Yes, 3 values
Data Normalization	Yes

The solution that was extracted using our system, although it achieves a 72.72% (16/22) accuracy in unknown data, it additionally enables the interpretation of the solution. The accuracy in the training data was 74.24% (49/66).

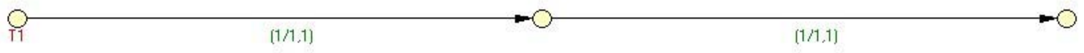
Table V. Input features of the Postoperative data.

Variable	**Feature**	**Values / Value range**
T1	L-CORE (internal patient temperature)	High (> 37), Medium (>= 36 and <= 37), Low (< 36)
T2	L-SURF (surface patient temperature)	High (> 36.5), Medium (>= 36.5 and <= 35), Low (< 35)
T3	L-O2 (oxygen saturation in %)	Excellent (>= 98), Good (>= 90 and < 98), Fair (>= 80 and < 90), Bad (< 80)
T4	L-BP (τελευταία μέτρηση πίεσης αίματος)	High (> 130/90), Medium (<= 130/90 and >= 90/70), Low (< 90/70)
T5	SURF-STBL (surface patient temperature stability)	stable, mod-stable, unstable
T6	CORE-STBL (internal patient temperature stability)	stable, mod-stable, unstable
T7	BP-STBL (blood pressure stability)	stable, mod-stable, unstable
T8	COMFORT (patient's perceived comfort at discharge)	0-20

From the solution that is shown in *Fig. 5*, we may extract the following simple rule system:

Q $\Leftarrow$ **conjunction** (Internal patient temperature)

The above rule can be freely interpreted into a simple rule statement, like "*the higher is Internal patient temperature, the more likely is to keep the patient in the hospital*".



```
(CNLN (In T1) (Rule 0 0))
```

Fig. 5. Solution description and the corresponding neural logic network for the Postoperative problem.

Diagnosis of Breast Cancer

This database of Breast cancer patients has been created in the Medical University of Winsconsin [27,28]. The diagnosis is concerned with the classification of a tumor as benign or malignant. In the past, a part of this data has been investigated using linear programming [29,30] and the accuracy in a total of 169 records varies between 93.5% and 95.9% depending on the system tuning. In a later data set we investigate (369 records in total), they have been applied machine learning techniques [31]. The accuracy, in a test set of 169 records, varies between 92.2% and 93.7% depending on the system tuning. In the full data that are available today (699 records), 458 (65.5%) concern benign tumors and the rest 241 (34.5%) malignant ones. The missing data percentage is low (1 ‰).

Table VI. Domain description for the Breast cancer diagnosis.

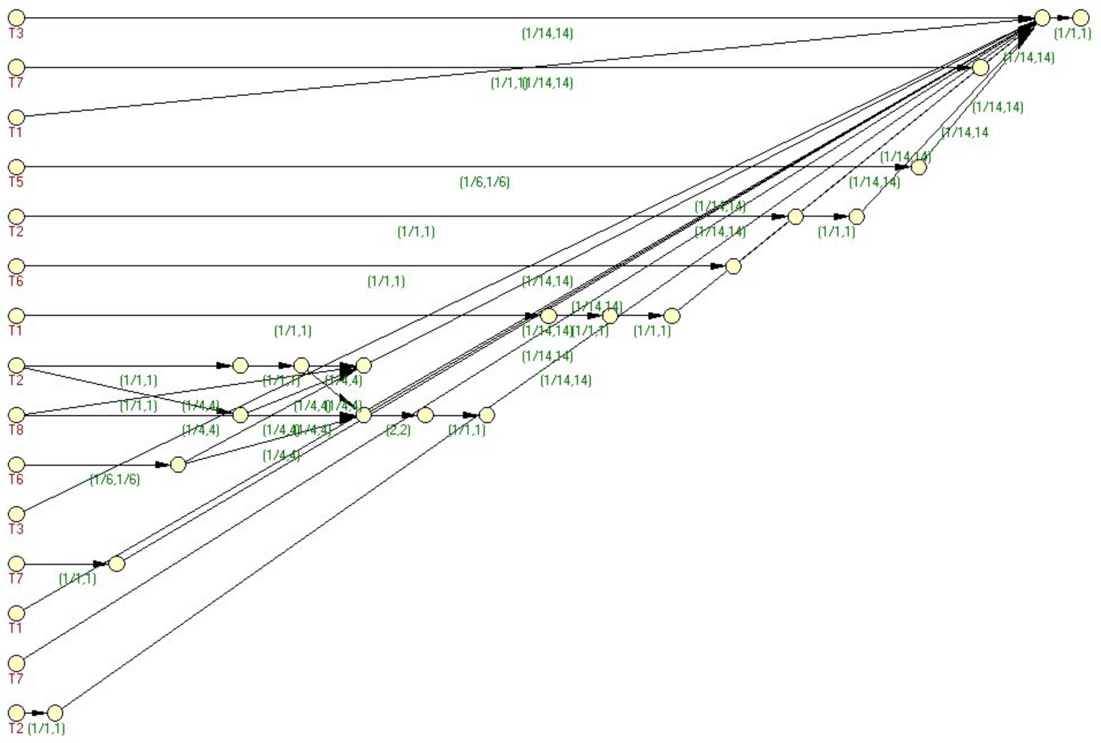
Domain	Diagnocis of Breast cancer
Data Base	Cancer
Input Features	9
Continuous	9
Discrete	0
Binary	0
Anti-Overfitting Method	Validation Set
Total Records	696
Training Set Records	348
Validation Set Records	174
Test Set Records	174
Missing Data	Yes, 16 values
Data Normalization	Yes

The database features are laboratory measurements of the cells. More specifically, these features are presented in *Table VII*. To avoid overfitting during training we made use of a validation set.

Table VII. Input features of the Breast cancer database.

Variable	**Feature**	**Values / Value range**
T1	Clump Thickness	1-10 -> 0..1
T2	Uniformity of Cell Size	1-10 -> 0..1
T3	Uniformity of Cell Shape	1-10 -> 0..1
T4	Marginal Adhesion	1-10 -> 0..1
T5	Single Epithelial Cell Size	1-10 -> 0..1
T6	Bare Nuclei	1-10 -> 0..1
T7	Bland Chromatin	1-10 -> 0..1
T8	Normal Nucleoli	1-10 -> 0..1
T9	Mitoses	1-10 -> 0..1

The extracted solution is shown in *Fig. 6*. This solution achieved accuracy 94.25% (164/174) in unknown data (test set). The solution was extracted after 26,000 iterations and the accuracy in the training set and in the validation set was 97.99% (341/348) and 97.12% (169/174) correspondingly. Although this solution is consisted of a relatively large neural logic network, it still maintains its interpretability a (large) number of logic rules. In this solution, only 7 out of 8 features are used. Specifically, the variable of 'Marginal Adhesion' is not used.



```
(CNLN (P1 (P1 (In T3) (S1 (In T7) (Rule 0 0) E)) (P1 (P1 (P1 (P1 (P1 (In T1) (S1 (In T5) (Link 257 2
(Rule 6 6)) E)) (S1 (S1 (In T2) (Link 115 0 (Link 179 0 (Rule 0 0))) E) (Link 115 0 (Rule 0 0)) E))
(S1 (In T6) (Rule 0 0) E)) (P1 (S1 (In T1) (Rule 0 0) (S2 E (Rule 0 0) (S2 E (Rule 0 0) E))) (S1 (P1
(S1 (In T2) (Rule 0 0) (P2 (S2 E (Rule 0 0) E) (Rule 0 0) E)) (P1 (In T8) (S1 (In T6) (Rule 6 6)
E))) (Rule 0 0) (P2 E (Rule 0 0) (S2 (S2 E (Rule 2 4) E) (Rule 0 0) E))))) (P1 (P1 (In T3) (S1 (In
T7) (Rule 0 0) E)) (P1 (P1 (In T1) (In T7)) (S1 (In T2) (Link 115 0 (Rule 0 0)) E))))) (Rule 6 6))
```

Fig. 6. Solution description and the corresponding neural logic network for the Breast cancer diagnosis problem.

CONCLUSIONS AND FURTHER RESEARCH

Neural networks are powerful connectionist systems that have been introduced in areas where symbolic processing systems of traditional artificial intelligence used to be applied. As a tool of computational intelligence, the adaptation of the neural network to the problem domain using an inductive method, offers advantage over expert systems where the knowledge must be acquired first, before the system development. Ever since their first application, interpretation of the obtained knowledge was a research target for neural networks. In the scope of this area, the neural logic networks have been proposed as a class of networks that by their definition preserve their interpretability into symbolic knowledge.
Until recently however, the application of an effective training / production method within the CI framework has not been successful. A novel system that uses genetic programming with indirect encoding that has been proposed recently [4], overcomes these problems, producing automatically designed and tuned neural logic networks, which always preserve their interpretability. In this work we applied the system into two real-world medical problems, the Postoperative treatment diagnosis and the diagnosis of the Breast Cancer. In both problems, the system has been proved capable of producing competitive to the literature results, which maintain their interpretability and lead to potential knowledge extraction.
Future work involves the application of the system in other areas, as well as the incorporation of recursive structures into the neural logic network architecture. Moreover, the minimum description length principle will be developed to be included as an anti-overfitting measure into the Breast cancer problem. Finally, parameter-tuning optimization of the underlying genetic programming algorithm is expected to offer better efficiency, hence it will be of primary importance among our future work.

REFERENCES

[1] Teh H.H., *Neural Logic Networks: A New Class of Neural Networks*, World Scientific Pub Co, 1995.

[2] Tan A-H. and Teow L-N., “Inductive neural logic network and the SCM algorithm”, *Neurocomputing*, Vol. 14, 2 : 5, pp.157-176, 1997

[3] Chia H.W-K. and Tan C-L.,"Neural logic network learning using genetic programming*", Intl. Journal of Comp. Intelligence and Applications*, 1:4, 2001, pp 357-368.

[4] Tsakonas A., Aggelis V., Karkazis I. and Dounias G., “An Evolutionary System for Neural Logic Networks using Genetic Programming and Indirect Encoding”, *Journal of Applied Logic*, accepted for publication, Elsevier, Spring 2004.

[5] Alba E., Cotta C. and Troya J.M., "Evolutionary Design of Fuzzy Logic Controllers Using Strongly-Typed GP", *Proc. 1996 IEEE Int'l Symposium on Intelligent Control*, pp. 127-132. New York, NY., 1996.

[6] Tsakonas A., Dounias G., "Hierarchical Classification Trees Using Type-Constrained Genetic Programming*", Proc. of 1st Intl. IEEE Symposium in Intelligent Systems*, Varna, Bulgaria, 2002.

[7] Tsakonas A., Dounias G., Axer H., and von Keyserlingk D.G., "Data Classification using Fuzzy Rule-Based Systems represented as Genetic Programming Type-Constrained Trees", *Proc. of the UKCI-01*, Edinbourgh, UK, pp 162-168, 2001.

[8] Tsakonas A. and Dounias G., "A Scheme for the Evolution of Feedforward Neural Networks using BNF-Grammar Driven Genetic Programming", *Proc. of Eunite-02*, Algarve, Portugal, 2002.

[9] Yu T. and Bentley P., "Methods to Evolve Legal Phenotypes", *Lecture Notes in Comp. Science 1498, Proc. of Parallel Problem Solving from Nature V*, pp 280-291,1998.

[10] Gruau F., Whitley D. and Pyeatt L., "A Comparison between Cellular Encoding and Direct Encoding for Genetic Neural Networks", in Koza J.R., Goldberg D.E., Fogel D.B., Riolo R.L., Eds.*,, Genetic Programming 1996: Proceedings of the First Annual Conf.*, pp 81-89, Cambridge, MA, MIT Press, 1996.

[11] Montana D.J., "Strongly Typed Genetic Programming", *Evolutionary Computation*, vol. .3, no. 2, 1995.

[12] N.Paterson and M.Livesey,"Evolving Caching Algorithms in C by GP", *Genetic Programming 1997*, pp 262-267, MIT Press, 1997.

[13] Naur P., "Revised report on the algorithmic language ALGOL 60", *Commun. ACM*, Vol 6, No 1, pp 1-17, Jan 1963.

[14] Whigham P., "Search Bias, Language Bias and Genetic Programming", *Genetic Programming 1996*, pp 230-237, MIT Press, 1996

[15] Gruau F., "Neural Network Synthesis using Cellular Encoding and the Genetic Algorithm", *Ph.D. Thesis*, Ecole Normale Superieure de Lyon, anonymous ftp:lip.ens-lyon.fr (140.77.1.11) pub/Rapports/ PhD PhD94-01-E.ps.Z.

[16] Gruau F., "On Using Syntactic Constraints with Genetic Programming", in P.J.Angeline, K.E.Jinnear,Jr., Eds., *Advances in Genetic Programming,* MIT,1996.

[17] Tsakonas A. and Dounias G., Decision Making in the Medical Domain: Comparing the Effectiveness of GP-Generated Fuzzy Intelligent Structures, *Proc. of Eunite-03*, Oulou, Finland, 2003.

[18] Wong M.L., "A flexible knowledge discovery system using genetic programming and logic grammars", *Decision Support Systems*, 31, 2001, pp 405-428.

[19] Hussain T., "Cellular Encoding: Review and Critique", *Technical Report, Queen's University*, 1997, http://www.qucis.queensu ca/home/hussain/web/1997_cellular_encoding_review.ps.gz

[20] J.Koza, F. Bennett, D. Andre and M. Keane, *Genetic Programming III: Automatic Programming and Automatic Circuit Synthesis*, Morgan Kaufmann, 2003.

[21] Ratle A. and Sebag M., "Genetic Programming and Domain Knowledge: Beyond the Limitations of Grammar-Guided Machine Discovery", Schienauer et al., Eds., *Proc. of the 6th Conf. on Parallel Problems Solving from Nature, LNCS*, Springer, Berlin, 2000, pp 211-220

[22] Singleton A., "Genetic Programming with C++", *BYTE Magazine*, Feb 1994.

[23] Smith,J.W., Everhart,J.E., Dickson,W.C., Knowler,W.C., and Johannes,R.S., "Using the ADAP learning algorithm to forecast the onset of diabetes mellitus", *Proc. of the Symposium on Computer Applications and Medical Care*, pp. 261--265. IEEE Computer Society Press, 1988.

[24] Diaconis,P. and Efron,B. "Computer-Intensive Methods in Statistics", *Scientific American,* Vol. 248, 1983.

[25] A. Budihardjo, J. Grzymala-Busse, L. Woolery. Program LERS_LB 2.5 as a tool for knowledge acquisition in nursing, Proceedings of the 4th Int. Conference on Industrial & Engineering Applications of AI & Expert Systems, pp. 735-740, 1991

[26] L. Woolery, J. Grzymala-Busse, S. Summers, A. Budihardjo .The use of machine learning program LERS_LB 2.5 in knowledge acquisition for expert system development in nursing. Computers in Nursing 9, pp. 227-234, 1991.

[27] O. L. Mangasarian and W. H. Wolberg, Cancer diagnosis via linear programming, SIAM News, Volume 23, Number 5, September 1990, pp 1 - 18.

[28] O. L. Mangasarian, R. Setiono, W.H. Wolberg, Pattern recognition via linear programming: Theory and application to medical diagnosis", in: "Large-scale numerical optimization, Thomas F. Coleman and Yuying Li, editors, SIAM Publications, Philadelphia 1990, pp 22-30.

[29] W.H. Wolberg and O.L. Mangasarian, Multisurface method of pattern separation for medical diagnosis applied to breast cytology, Proceedings of the National Academy of Sciences, U.S.A., Volume 87, December 1990, pp 9193-9196.

[30] K. P. Bennett, O. L. Mangasarian: "Robust linear programming discrimination of two linearly inseparable sets", Optimization Methods and Software 1, 1992, 23-34,Gordon & Breach Science Publishers.

[31] Zhang,J., Selecting typical instances in instance-based learning. In Proceedings of the Ninth International Machine Learning Conference, pp. 470-479, 1992, Aberdeen, Scotland: Morgan Kaufmann.